VOUGHT MISSILES
AND SPACE COMPANY
TEXAS DIVISION

P. O. Box 6267 Dallas, Texas 75222

CR-128593

Performed Under
NASA-MSC Contract
NAS9-4776

N73-10963

MODULAR THERMAL ANALYZER ROUTINE

Report No. 00.1524

Volume I

27 March 1972

Submitted by

Vought Missiles and Space Company
LTV Aerospace Corporation
P.O. Box 6267
Dallas, Texas

to

National Aeronautics and Space Administration
Manned Spacecraft Center
Houston, Texas

Prepared by:

J. A. Oren

M. A. Phillips

D. R. Williams

Approved by:

R. J. French, Supervisor
Environmental Control/
Life Support Systems

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF APPENDICES

* Contained in Volume II

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

The Modular Thermal Analyzer Routine (MOTAR) is a general thermal analysis routine with strong capabilities for performing thermal analysis of systems containing flowing fluids, fluid system controls (valves, heat exchangers, etc.), life support systems, and thermal radiation situations. Its modular organization permits the analysis of a very wide range of thermal problems from simple problems containing a few conduction nodes to those containing complicated flow and radiation analysis with each problem type being analyzed with peak computational efficiency and maximum ease of use.

MOTAR gives its user the ability to obtain the transient or steady state solution of a problem using either the forward differencing, mid-differencing or backward differencing finite difference solution methods. Transient and steady state analyses may be performed during the course of a single problem so that as an example, a transient analysis may be initiated at some steady state condition. In addition any number of transient and/or steady state problems may be analyzed on a given problem by applying the user logic capability on MOTAR.

Numerous options are available with MOTAR for determining time and temperature dependent thermal conductors, capacitances, and absorbed heat values. In addition to these standard options the user may supply any functional relationship desired for these elements in the user logic block. Option are available for analysis of convection and flow conductors which utilizes the results of the simultaneously performed flow analysis. Also, extensive radiation analysis capability is supplied which provides for determination of radiation interchange factors for any combination of specular and diffuse radiation. A number of thermal nodes may be combined into a single surface to greatly reduce the amount of computer time required for determining the interchange factors and calculating the net heat flow due to radiation interchange during the problem. Additional thermal analysis capabilities available as options include (1) cabin air thermal and mass balance analysis including condensation and/or evaporation from the walls (2) heat exchanger simulation ability for counterflow, crossflow, and parallel flow exchangers and (3) inline heater analysis.

A pressure/flow analysis of a fluid flow system consisting of an arbitrary tube network may be performed simultaneous with the thermal analysis so that on each iteration the thermal problem is updated based upon the latest flow conditions and vice versa. If only a pressure/flow analysis is desired with no thermal analysis this may also be performed on MOTAR. The tube/flow path methods used on previous VMSC routines for flow analysis have been replaced in MOTAR with a pressure node/tube conductor network method. Using this method the number of simultaneous equation that must be solved is reduced. Also, the user has much more flexibility as to the type of flow system that he may analyze. For instance, the user may connect any number of tubes at a junction from two to a large number, whereas previously three and only three

connections were permitted. Also, the connections need not follow any fixed pattern as before (i.e., off-center flow paths are handled automatically). The valve pressure drops are readily included in the pressure/flow balance for all types of valves with the revised method contrary to previous methods. Several options are available for obtaining friction factors and head loss values.

To enhance the pressure/flow analysis capability extensive valve and pump analysis capabilities have been included in MOTAR. The previous five valve types plus considerable additional capabilities have been simplified into three valve types. The valves have been formulated so that the control of either cooling (space radiator) or heat (solar absorber) situations may be controlled with any of the valve types. Pump options including tabulated pump curve of pump flow vs pressure drop or polynomial curves of pressure drop vs flow rate are available with accelerated methods for convergence of the pump and flow system characteristics.

The input for MOTAR has been designed to give the user a high degree of effectiveness and flexibility while maintaining an easy-to-use format. The effectiveness is accomplished by providing powerful options to the user which permit the input of large quantities of data with a single entry in the input. The flexibility is obtained by providing the user with a large number of options for each data entry. Many features were incorporated to make the routine easy to use which include the use of descriptive names to identify data blocks, the ability to omit blocks not requiring input data for a given problem, and the use of a free form input format. Several input/output options were made available to aid the user which take advantage of the magnetic tapes available on the Univac 1108 computer. Included in these were the data tape/edit capability, restart tape ability, plotting and/or starting from a previously generated history tape and the supplying of heat flux values on a tape.

The organization and programming methods applied to MOTAR achieved a high degree of computer utilization efficiency in terms of computer execution time and storage space required for a given problem. The computer time required to perform a given problem on MOTAR is approximately 40 to 50 percent that required for the currently existing widely used routines [14,15,24*]. The computer storage requirement for MOTAR is approximately 25 percent more than the most commonly used routines [14,15] for the most simple problems but the data storage techniques for the more complicated options should save a considerable amount of space.

---

* Superscripts represent the reference number for references found in Section 6.0.

2.0 INTRODUCTION

During the past eight years the Vought Missiles and Space Company (VMSC) of LTV Aerospace Corporation has been involved in the development of special purpose computer routines for the thermal analysis of systems with flowing fluid, life support components and enclosure radiation. The development of these special purpose routines was necessary to obtain the analytical capability required for the design and simulation of space radiator systems [12,19,20] environmental control and life support systems [16,21] and fuel cell cooling systems, [22,23] since the required analytical tools didn't exist. Some of the VMSC developed routines were used for thermal simulation of the entire spacecraft for the LM ascent stage, [16] LM decent stage, [24] and the Apollo Block I and Block II Command Modules [22,25].

Advances in computer technology such as the advent of the Univac 1108 Fortran V computer language made possible the assembling of the previously developed specialized capabilities into a single computer routine without loss of computational efficiency. This assembly of these capabilities into a user oriented routine efficient in terms of both computer time and space was the objective of the Modular Thermalizer Routine (MOTAR) which was developed by VMSC and is described in this report.

3.0 ROUTINE ANALYTICAL METHODS

This section describes the analytical methods used in the Modular Thermal Analyzer Routine (MOTAR). MOTAR solves thermal and flow problems simultaneously for either transient or steady state temperature conditions. In either case, it solves a discrete lumped parameter finite difference user input Mathematical Model. Several methods of solution are available to the user for the transient thermal problem including explicit forward differencing, implicit mid-differencing and implicit backward differencing. Two methods are available for solution of the flow problem; namely, the direct solution method using the Gauss-Jordan elimination technique for small problems, and the successive over relaxation method for larger problems.

MOTAR contains special capabilities to enhance thermal and flow analysis. Capabilities related to thermal analysis of flowing and life support systems include the calculation of flow and convection thermal conductors using flow data, numerous options for determining heat exchanger performance, and cabin thermal and mass analysis. In addition to the normal radiation conductor capability, provisions are available for determination of radiation interchange factors (Script - F) for any combination of specular and diffuse radiation and incorporation of those factors into the thermal analysis. These Script F values can be determined for infrared as well as any number of wave length bands for incoming non-infrared radiation. The pressure flow analysis is augmented with numerous options for determining friction factors, determining valve performance, and balancing the system flow rates with pump curves.

These capabilities are discussed in detail in Sections 3.1 and 3.2 which follow. Thermal analysis capabilities are discussed in 3.1 and pressure/flow analysis capabilities are discussed in Section 3.2.

3.1 THERMAL ANALYSIS

MOTAR gives its user the capability to determine the approximate solution to the differential equations which govern the transient temperature behavier in a media. This solution is obtained by approximating the non-linear partial differential equations with a set of difference equations which are solved by successively solving a set of algebraic equations. Provisions have been incorporated into the solution methods to permit the analysis of a general n-dimensional structural problem as well as problems containing radiation and convection to a fluid. Because the finite difference method evaluates equation constants at very short time intervals nonlinearities such as radiation, varying convection coefficients, and temperature dependent properties are easily approximated by linearizing over the small intervals. Numerous options are available for evaluation of these nonlinearities. Options are also available for characterizing components commonly found in flowing fluid and life support systems such as heat exchangers, fluid heaters and life support cabins.

The following sections describe the MOTAR thermal analysis capability in detail. A brief derivation of the finite difference equations is given in Section 3.1.1. The methods for solving these equations are discussed in

3.1.2 and the supporting features for evaluating the nonlinear coefficients and characterizing the special components are discussed in Section 3.1.3.

3.1.1 Finite Difference Analysis

The nonlinear partial differential equation which governs the heat conduction in a three dimensional conducting media can be derived by use of Fouriers heat conduction equation and an energy balance on a differential element to be

$$\rho C_p \frac{\partial T}{\partial \tau} = \frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial T}{\partial z}\right) + q \qquad (1)$$

where T = the temperature which is a function of x, y, z and time, τ

ρ = density of the material

C_p = specific heat of the material

kx, ky, kz = thermal conductivity in directions x, y, and z

q = the heat generation per unit volume and unit time

The thermal properties, ρ, C_p, and k, can in general be functions of temperature which makes equation (1) a second order, nonlinear partial differential equation. A general form of the boundary conditions which may be encountered is given by

$$f_1(X_B,Y_B,Z_B,T) \left.\frac{\partial T}{\partial N}\right|_{X_B,Y_B,Z_B} =$$

$$f_2(X_B,Y_B,Z_B,\tau,T) * T(X_B,Y_B,Z_B) + f_3(X_B,Y_B,Z_B,\tau,T) \qquad (2)$$

where X_B,Y_B,Z_B is the body surface

N is the surface normal

When $f_1=0$, the boundary condition is that of spatially variant and time variant surface temperature. When $f_2 = 0$, the boundary condition is that of spatially variant and time variant surface heat flux. For f_1, f_2 and f_3 all non-zero, the equation represents a convective, radiative, or combined convective/radiative boundary condition where both the heat transfer coefficient and the sink temperature are spatially variant and time variant.

The equations defined by equations (1) and (2) have been solved for a large number of special cases [1-10] but in general approximations must

be introduced in order to solve a given problem. One such approximation which permits the general solution of equations (1) and (2) (subject to the approximation) is that of finite difference representation of the differential quantities. This permits the non-linear partial differential equation given by (1) to be written in terms of a set of linear algebriaic equations which with the aid of the electronic computer can be successively solved to obtain general problem solution. This is the approach taken in MOTAR.

Using the finite difference method, the partial derivatives given by equation (1) are approximated by differences as follows:

Let

$$\frac{\partial T}{\partial \tau} = \frac{T_i^{n+1} - T_i^{n}}{\Delta \tau} \tag{3}$$

where T_i^{n+1} = the temperature at node i at the finite approximation of time (iteration) n + 1. Node i is located at x, y, z

T_i^{n} = the temperature of node i at time n

$\Delta \tau$ = the finite time increment

We will assume the thermal properties can be assumed constant over the small time increment, $\Delta \tau$. Thus, the terms on the right side of equation (1) become

$$k_n \frac{\partial^2 T}{\partial N^2}$$

where N is either x, y, or z

We will approximate $\frac{\partial^2 T}{\partial N^2}$ by subdividing each N coordinate into a grid in space of width ΔN. The approximation is then,

$$\frac{\partial^2 T}{\partial N^2} = \frac{\frac{T_{i-\Delta n}^{m} - T_i^{m}}{\Delta N} - \frac{T_i^{m} - T_{i+\Delta n}^{m}}{\Delta N}}{\Delta N} \tag{4}$$

$$= \frac{T_{i-\Delta n}^{m} - T_i^{m}}{\Delta N^2} + \frac{T_{i+\Delta n}^{m} - T_i^{m}}{\Delta N^2}$$

Where

N	is the direction of the derivative (x,y,or z)
$T_{i-\Delta n}^{m}$	is the temperature of the node adjacent to i in the -N direction at time m
T_i^m	is the temperature of node i at time m
$T_{i+\Delta n}$	is the temperature of the node adjacent to i in the +N direction at time m
ΔN	is the grid with in the N direction
m	is the time for evaluation of the derivative somewhere between τ and $\tau + \Delta\tau$.

If the above approximations of equations (3) and (4) are substituted into equation (1) we get

$$\rho_i C_{pi}\left(\frac{T_i^{n+1}-T_i^n}{\Delta\tau}\right) = k_x\left[\frac{T_{i-\Delta x}^m - T_i^m}{\Delta X^2} + \frac{T_{i+\Delta x}^m - T_i^m}{\Delta X^2}\right]$$

$$+ k_y\left[\frac{T_{i-\Delta y}^m - T_i^m}{\Delta y^2} + \frac{T_{i+\Delta y}-T_i^m}{\Delta y^2}\right] + k_z\left[\frac{T_{i-\Delta z}^m - T_i^m}{\Delta z^2} - \frac{T_{i+\Delta z}^m - T_i^m}{\Delta z^2}\right]$$

$$+ q$$

If we multiply the above equation by the volume of node i, ΔV_i , where

$$\Delta V_i = \Delta X \cdot \Delta y \cdot \Delta z$$

We get

$$\rho_i \Delta V_i C_{pi}\left(\frac{T_i^{n+1}-T_i^n}{\Delta\tau}\right) = \frac{k_x \Delta y \Delta z}{\Delta X}\left(T_{i-\Delta x}^m - T_i^m\right)$$

$$+ \frac{k_x \Delta y\ \Delta z}{\Delta X}\left(T_{i+\Delta x}^m - T_i^m\right) + \frac{k_y \Delta X \Delta Z}{\Delta Y}\left(T_{i-\Delta y}-T_i\right) + \frac{k_y \Delta X \Delta Z}{\Delta Y}\left(T_{i+\Delta y}-T_i\right)$$

$$+ k_z \frac{\Delta X\ \Delta Y}{\Delta Z}\left(T_{i-\Delta z}^m - T_i^m\right) + k_z \frac{\Delta X \Delta Y}{\Delta Z}\left(T_{i+\Delta z}-T_i^m\right) + q_i \Delta V_i$$

If we note that

$\rho_i \Delta V_i = W_i$ = Mass of node i

$\Delta y \cdot \Delta z = A_x$ = Area for conduction in the x direction

$\Delta x \cdot \Delta z = A_y$ = Area for conduction in the y direction

$\Delta x \cdot \Delta y = A_z$ = Area for conduction in the z direction

$q \Delta v = Q_i^m$ = The total heat originating at node i at time m

then we can write the above equation as

$$W_i C_{pi} \left(\frac{T_i^{n+1} - T_i^n}{\Delta \tau} \right) = \frac{K_x A_x}{\Delta X} \left(T^m_{i-\Delta x} - T_i^m \right) + \frac{k_x A_x}{\Delta X} \left(T^m_{i+\Delta x} - T_i^m \right)$$

$$+ \frac{k_y A_y}{\Delta Y} \left(T^m_{i-\Delta y} - T_i^m \right) + \frac{K_y A_y}{\Delta Y} \left(T^m_{i+\Delta y} - T_i^m \right) + \frac{k_z A_z}{\Delta Z} \left(T^m_{i-\Delta z} - T_i^m \right)$$

$$+ \frac{k_z A_z}{\Delta z} \left(T^m_{i+\Delta z} - T_i^m \right) + Q_i^m \qquad (5)$$

If we examine equation (5) term by term we see that the left side is the rate of heat storage required to raise the temperature of the node i mass, W_i, at the rate of $(T_i^{n+1} - T_i^n)/\Delta\tau$ and is thus the heat storage rate of node i. Each of the first six terms on the right of equation (5) represent the rate of heat transfer into node i at one of the six surfaces of the three dimensional parallelepipid from surrounding nodes. The $kA/\Delta X$ portion of these terms represent the thermal conductance between node i and the adjacent node j. Q_i represents the total rate of heat originating at node i. Thus, the right hand side represents the total heat transfer rate into node i from its surroundings. If we define

$$U_{ij} = \left(\frac{kA}{\Delta x} \right)_{ij}$$

and

$$C_i = W_i C_{pi}$$

We can write equation (5) in the general form

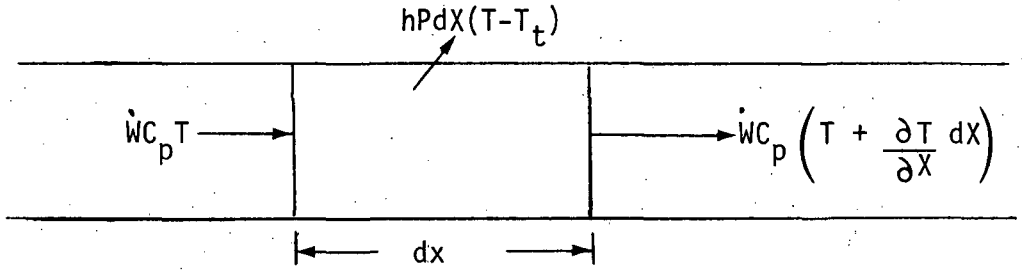
$$C_i \left(\frac{T_i^{n+1} - T_i^n}{\Delta \tau} \right) = \sum_{j=1}^{nc} U_{ij} \left(T_j^m - T_i^m \right) + Q_i^m \qquad (6)$$

where nc is the number of conductances attaching node i to surrounding nodes

C_i is the thermal capacitance of node i

U_{ij} is the thermal conductance between nodes i and j.

Equation (6) was defined in terms of conduction heat transfer only. It can be easily extended to analyze a flowing fluid. Consider fluid flowing in a tube. The energy balance on an elemental length, dx , of the fluid can be used to derive the governing differential equation



Energy Stored = Energy in - energy out

$$\rho A_c dX C_p \frac{\partial T}{\partial \tau} = \dot{W}C_p T - \dot{W}C_p\left(T + \frac{\partial T}{\partial X} dX\right) - hPdX\,(T-T_t)$$

or

$$\rho A_c C_p \frac{\partial T}{\partial \tau} = -\dot{W}C_p \frac{\partial T}{\partial X} - hP(T-T_t) \qquad (7)$$

where ρ = fluid density

A_c = Fluid flow cross sectional area

C_p = Fluid specific heat

τ = time

$\dot{W}$ = fluid mass flow rate down tube

P = wetted perimeter of flow passage

T = fluid temperature

T_t = tube temperature

h = convection heat transfer coefficient

If we make the following difference approximations for the partial derivatives,

$$\frac{\partial T}{\partial \tau} = \frac{T^{n+1} - T^{n}}{\Delta \tau}$$

$$\frac{\partial T}{\partial X} = \frac{T^{m} - T^{m}_{x-\Delta x}}{\Delta X}$$

substitute these into equation (7) and multiply by ΔX we get the finite difference fluid heat balance equation for an element of length ΔX and cross sectional area A_c

$$\rho A_c \Delta X \, C_p \left[\frac{T^{n+1} - T^{n}}{\Delta \tau}\right] = -\dot{W}CP \left[T^{m} - T^{m}_{x-\Delta x}\right] - hP\Delta X \left[T^{m} - T_t{}^{m}\right]$$

or

$$C \left[\frac{T^{n+1} - T^{n}}{\Delta \tau}\right] = \dot{W}C_p \left[T_u{}^{m} - T^{m}\right] + hA_{ht} \left[T_t{}^{m} - T^{m}\right] \tag{8}$$

where T^{n+1} = fluid temperature of the node at location X at time $\tau + \Delta\tau$

T^{n} = fluid temperature at X and

$T_u{}^{m}$ = fluid temperature of node at and time τ if $m = n$ and $\tau + \Delta\tau$ if $m = n + 1$

T^{m} = fluid temperature of node at X and time τ if $m = n$ or at time $\tau + \Delta\tau$ if $m = n + 1$

$T_t{}^{m}$ = tube temperature of tube node at X and time τ if $m = n$ or at time $\tau + \Delta\tau$ if $m = n + 1$

C = $\dot{W}C_p$ product for the fluid node

Note that equation (8) can be cast in the same form as equation (6) with one exception. The energy represented by the $\dot{W}C_p$ (Tu - T) term only flows one direction. That is, energy flows from the upstream fluid node but not from the downstream fluid node. Thus, if we make $\dot{W}C_p$ a one way conductor and the hA convection conductor, equation (8) is in the form of equation (6) and we see that we can extend equation (6) to include analysis of flowing fluid and convection.

Equation (6) can also be extended to include analysis of radiation heat exchange. If we define UA_{ij} by

$$UA_{ij} = \sigma \mathcal{F} A_{ij} \left[T_j^2 + T_i^2\right]\left[T_j + T_i\right]$$

where σ = the Stephan-Baltzman Constant

$\mathcal{F} A_{ij}$ = the radiation interchange factor

where in this case, UA_{ij} is the heat transfer coefficient between two surfaces by radiation then radiation heat transfer analysis can be included in equation (6). Thus, with the above definitions for conductances, equation (6) is applicable for a very wide class of thermal problems which include conduction radiation, convection and flowing fluid.

The linear algebriac equation (6) represents an approximation to the nonlinear second order partial differential equation (1). To obtain the general solution of temperature as a function of time and location in a body, equation (6) must be written for each nodal point in the body. (The shorter the distance between nodal points, the more accurate the solution.) These equations must then be solved simultaneously for values of temperature at the node locations at time $\tau + \Delta\tau$ based upon the temperature at time τ and the heat flow rate during the time between time τ and $\tau + \Delta\tau$. The time is then incremented so that τ becomes $\tau + \Delta\tau$ and the process repeated. Thus, the simultaneous equations are successively solved to obtain the temperature vs time for each nodal location in the problem. The approximate solution to equation (1) is then obtained in this manner.

As previously mentioned, the value of m in equation (6) represents the point within the time increment from τ to $\tau + \Delta\tau$ for evaluating the flow of heat. The choice of m has a significant effect on the problem solution formulation. For instance m = n, equation (6) becomes

$$C_i \left[\frac{T_i^{n+1} - T_i^n}{\Delta\tau}\right] = \sum_{j=1}^{nc} U_{ij}\left[T_j^n - T_i^n\right] + Q_i^n \qquad i=1, NN$$

and it can be solved explicitly for T^{n+1} in terms of known conditions at time n as follows:

$$T_i^{n+1} = T_i^n + \frac{\Delta\tau}{C_i}\left[\sum_{j=1}^{nc} U_{ij}\left(T_j^n - T_i^n\right) + Q_i^n\right] \qquad i=1,NN \tag{9}$$

Here, NN = the number of nodes. This is the explicit or forward-difference finite difference method.

If m in equation (6) is n + 1, it can be re-written as

$$C_i\left(\frac{T_i^{n+1}-T_i^{n}}{\Delta\tau}\right)=\sum_{j=1}^{nc} U_{ij}\left(T_j^{n+1}-T_i^{n+1}\right)+Q_i^{n+1}$$

or

$$\left(\frac{C_i}{\Delta\tau}+\sum_{J=1}^{nc} U_{ij}\right)T_i^{n+1}-\sum_{J=1}^{nc} U_{ij}T_j^{n+1}=T_i^{n}+Q_i^{n+1} \qquad i=1,NN \tag{10}$$

This equation represents a set of NN simultaneous equations to be solved for the T's and is called the implicit backward difference method. It is much more difficult to solve than equation (9) but it is more efficient than (9) for certain types of problems because it has no stability restrictions on the time increment.

Equation (9) assumes the heat transfer rates are established at the time τ, the start of the temperature iteration whereas, equation (10) assumes the rates are established at time $\tau+\Delta\tau$. A third method assumes the heat transfer rate is some weighted average between that at τ and that at $\tau+\Delta\tau$. Let the net heat transfer rate Qnet be given by

$$Qnet = A\,Q_{net}^{n+1}+[1-A]\,Q_{net}^{n}$$

Then

$$C_i\left(\frac{T_i^{n+1}-T_i^{n}}{\Delta\tau}\right)=A\,Q_{net}^{n+1}+[1-A]\,Q_{net}^{n}$$

and

$$C_i\left(\frac{T_i^{n+1}-T_1^{n}}{\Delta}\right)=A\left(\sum_{J=1}^{nc} U_{ij}\left[T_j^{n+1}-T_i^{n+1}\right]+Q_i^{n+1}\right)$$

$$+\ (1-A)\left(\sum_{J=1}^{nc} U_{ij}\left[T_j^{n}-T_i^{n}\right]+Q_i^{n}\right)$$

or

$$\left(\frac{C_i}{\Delta\tau}+A\sum_{J=1}^{nc} U_{ij}\right)T_i^{n+1}-\sum_{J=1}^{nc} AU_{ij}T_j^{n+1}=\frac{C_iT_i^{n}}{\Delta\tau}+A\,Q_i^{n+1}$$

$$+\ (1-A)\left(\sum_{J=1}^{nc} U_{ij}\left[T_j^{n}-T_i^{n}\right]+Q_i^{n}\right) \qquad j=1,NN \tag{11}$$

Equation (11) represents the general form of the implicit equation. When $A = 1$ the backward difference equation given by equation (10) results. When $A = 0$, the forward difference equation given by (9) results. When $A = 0.5$ the solution method is called the implicit mid-difference.

In MOTAR, equation (9) is solved for the explicit method and equation (11) is solved for the implicit methods. Only values for A of 0.5 and greater are considered for the implicit methods because stability problems arise for values of a less than 0.5.

A discussion of the implementation of these methods is presented in the following section.

3.1.2 Temperature Solution Methods

In the previous section the nonlinear partial differential equation governing the temperature in a material was cast in the form of a set of linear algebraic equations by use of finite difference approximation. These equations are solved by MOTAR to obtain the approximate temperature vs time trace for each lump location (or the steady state temperature distribution for steady state problems). This is done by obtaining successive solutions of the equations at small increments of time with each time point solution depending on that of the previous time. Two basic methods are currently available to the MOTAR user for evaluating these transient equations depending on the point in the time interval that the flow of heat is assumed to occur. These are the explicit method wherein the heat flow rate is assumed to be that at the start of the time step and the implicit method where the heat flow can be evaluated anywhere between mean over the time step and the end of the time step. Each of the methods are discussed in more detail below. Methods for steady state analysis are also discussed.

3.1.2.1 Explicit Temperature Solution

When, in obtaining the solution to the finite difference temperature equations, the flow of heat is assumed to occur at the start of the iteration the updated temperature for each lump can be solved directly from known values of temperatures, heat fluxes, and coefficients and no simultaneous solution is required. The general form of the equation to be solved for each node is given by

$$T_i^{n+1} = T_i^n + \frac{\Delta r}{C_i} \left(\sum_{J=1}^{nc} U_{ij}^n \left[T_j^n - T_j^n \right] + Q_i \right) \qquad i=1,NN \qquad (12)$$

where T_i^{n+1} = the temperature of node i at iteration n + 1

T_i^{n} = the temperature of node i at iteration n

$\Delta\tau$ = the iteration time increment

C = the lunar capacitance = $m \cdot C_p$

C_p = the specific heat

U_{ij} = the conductance between nodes i and j

nc = the number of conductances

Q_i = the absorbed incident heat or the heat generated internally

NN = the number of nodes

Since all times on the right side of equation (12) are known at the start of each iteration T_i^{n+1} can be solved directly. This is performed in MOTAR by subroutine EXPLCT and its referenced subroutines.

The values used for U_{ij} must be obtained by different methods depending on the mechanism for heat transfer, (i.e., conduction, connection, flowing fluid, etc.). The methods used in MOTAR for determining U_{ij} for heat transfer mechanisms are discussed in section 3.1.3.

Equation (12) gives a stable solution as long as the time increment, $\Delta\tau$, meets the following requirement

$$\Delta\tau \leq \frac{C_i}{\sum_{J=1}^{nlc} U_{ij} + \sum_{J=1}^{nrc} 4\sigma A_{ij} T_i^3} \qquad (13)$$

Where nlc = Number of linear conductor
nrc = Number of radiation conductor

MOTAR has two options which the user may speicfy regarding the time increment. If option 1 is specified, the time increment will be "overriden" for those lumps having smaller maximum time increments than the problem increment. When this occurs, the maximum time increment given by equation (13) is substituted for $\Delta\tau$ in equation (12), resulting in a "steady state" solution for those nodes overriden. If option 2 is specified, the entire problem will be iterated at the value of the smallest maximum time increment given by equation (13). The user may specify maximum and minimum values of the problem iteration increment for this option and if the maximum time increment of some node is below the specified minimum iteration increment the problem will be terminated.

3.1.2.2 Implicit Temperature Solution

The finite difference transient temperature equations can be formulated so that the flow of heat is assumed to occur at some point during the iteration time step other than the start. With this formulation, the set of temperature equations must be solved simultaneously as shown below.

The general form of the finite difference temperature equation assuming the rate of heat flow is evaluated at the fraction, A, of the iteration is given by

$$\left(\frac{C_i}{\Delta\tau} + A\sum_{J=1}^{nc} U_{ij}\right) T_i^{n+1} - \sum_{\substack{J=1 \\ J\neq i}}^{nc} A\,U_{ij}T_j^{n+1} = (1-A)\left(\sum_{J=1}^{nc} U_{ij}\left[T_j^n - T_i^n\right] + Q^n{}_i\right)$$

$$+ A\,Q_i^{n+1} + \frac{C_i}{\Delta\tau} T_i^n \qquad i = 1,N \tag{14}$$

Where the symbols are as defined in equation (12). A is the fraction of the heat flux at the end of the time step and (1-A) is the fraction of the heat flux at its start; i.e., if A=.5, the flux is the average between that at the start and that at the end of the iteration resulting in the mid-differencing technique. If A=1.0 all the heat flow is assumed to occur at the end of the iteration resulting in the backward difference method.

If we difine the following,

$$b_{ii} = \frac{C_i}{\Delta\tau} + A\sum_{J=1}^{nc} U_{ij}$$

$$b_{ij} = -A\,U_{ij} \qquad i \neq j$$

$$C_i = (1-A)\sum_{J=1}^{nc} U_{ij}\left[T_j^n - T_i^n\right] + Q^n{}_i + A\,Q^{n+1}{}_i + \frac{C_i}{\Delta\tau} T_i^n$$

we can write N equations from equation (14) for the N lumps in the problem.

$$\sum_{J=1}^{N} b_{ij}\,T_j^{n+1} = C_i \qquad i = 1, N$$

This set of equations are solved for T_j^{n+1} in MOTAR at each time step using a modified version of the successive-point-overrelaxation method. The following is a summary of the procedure:

1. Assume an initial temperature array called T

2. Calculate time and temperature dependent constants

3. Call PRETMP

4. Calculate values of temperature from equation one lump at a time. Call this value T_i for the ith lump. The value is then modified by the equation $T_i = T_i + ORP\ (T_i - T_i)$, ORP being the overrelaxation parameter. The value of T_i is then compared to T_i. If $T_i - T_i$ is less than an input DTMXA (iteration limit) the iteration of this particular equation is temporarily suspended. The value of T_i is set equal to T_i following the comparison.

5. This procedure is continued until each lump's equation has been iterated until the error satsifies the tolerence; i.e., until $T_i - T_i$ is less than DTMXA for each lump, is tentatively achieved.

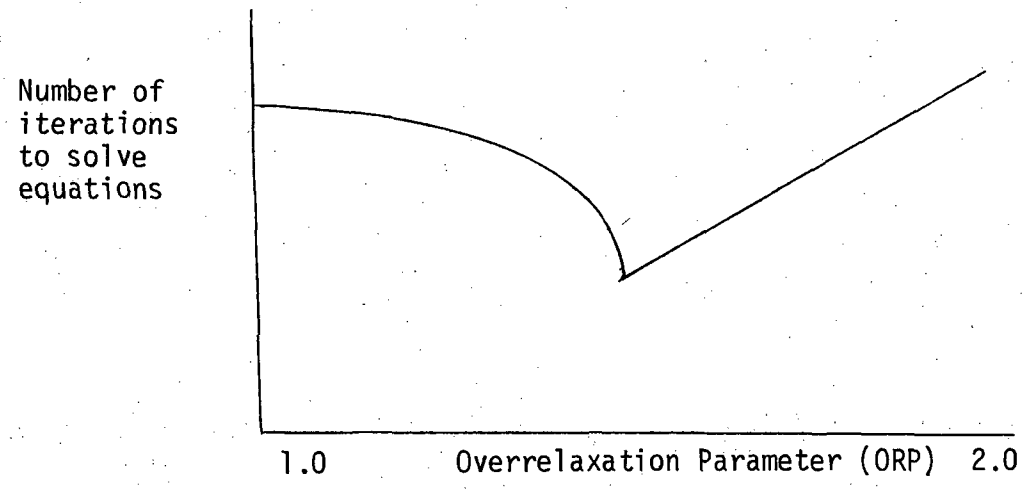
6. The process is repeated from step 3. As soon as the last lump satisfies the $T_i - T_i$ DTMXA, if all equations were not iterated, the process is again begun for each lump from step 3.

The standard SOR procedure is modified in that those equations which satisfy the $|T_i' - T_i| < DTMXA$ are not iterated until all equations have satisfied the relation. For some problems this procedure has some of the features of a block-iterative solution. If in a large problem a heat source is localized, the temperature change would, in an ordinary SOR method, propagate outward from the heat source in waves. With the above modification, only those equations of nodes in the immediate area of the distrubance will be iterated on the second and succeeding few iterations. After these reach the prescribed iteration limit the procedure is restarted and the number of nodes whose equations are iterated continuously increases. After a certain time subsequent to the passage of the "solution wave" through the various lumps, those lumps near the original distrubance will have reached a steady state value such that further iterations would not alter their temperature more than the prescribed limit applied to the iterations. This modification therefore intuitively leads to a fewer number of iterations, on the average.

The iteration limit, DTMXA, on the equations is assumed to be that which guarantees the error in the iteration process to be less than some specified amount. This amount of iteration error should be chosen to be well within the expected truncation error.

The selection of overrelaxation parameters, ORP, is of paramount importance to the user, in that proper selection of it can reduce run time significantly. Theoretical analyses of convergence rates for the successive overrelaxation iteration are summarized in Reference 3. For elliptic

equations with boundary conditions of the type that the function is specified constant on the boundary, a typical predicted curve of the number of iterations to solve the equation as a function of the overrelaxation parameter is as shown in the sketch below. A value of 1.4 has been found from experience to be a good first estimate.



3.1.2.3 Steady State Solution

Two methods are available in MOTAR for determining the steady state solution for the temperature distribution in a given problem. The first method iterates the basic explicit equation, equation (12), to solution. The second method obtains the solution to the implicit equation given by equation (14). The first method results in a block iterative Sidel iteration method and the second results in a point iterative Sidel method with successive overrelaxation. The second method generally converges faster but the first requires less space and thus, can handle the largest problem.

Explicit Steady State

The explicit steady state method is basically the solution of equation (12) with constant boundary conditions. To accelerate the convergence the maximum time increment, $\Delta\tau_{max}$, for each node given by equation (13) is substituted for $\Delta\tau$ in equation (12). This results in the steady state solution for each node with its surrounding conditions on the previous iteration for each iteration for problems with no radiation. For problems with radiation this results in the largest stable change permitted.

Implicit Steady State

The implicit steady state method is basically the solution of equation (14) with constant boundary conditions, with A=1 and with a large

value for $\Delta\tau$, the time increment, so that the terms $C/\Delta\tau$ approach zero. This results in the equation

$$T_i - \frac{\sum U_{ij} T_j}{\sum U_{ij}} = \frac{Q_i}{\sum U_{ij}} \qquad i=1,n \qquad (15)$$

Which is solved using the iterative procedure described in section 3.1.2.2.

3.1.3 Thermal Analysis Features

This section describes some of the more significant thermal analysis features used to enhance the temperature solution methods described in Section 3.1.2. The items discussed are (1) conductor calculation methods, (2) heat exchanger analysis method, (3) cabin analysis method and (4) radiation interchange methods.

3.1.3.1 Conductor Calculation Method

The values of U_{ij} in equations (12) and (14) are determined by different methods depending on the heat transfer mechanism. The methods used to determine the conductors for conduction, convection, radiation, and for fluid crossing the boundary from one node into another are described below.

Conduction

The conductance between two nodes for conduction heat transfer is given by:

$$U_{ij} = \frac{k A_{ij}}{\Delta X_{ij}}$$

Where k = thermal conductivity of lumps i and j
A_{ij} = the area for conduction between i and j
ΔX_{ij} = the conduction distance from the center of node i to the center of node j

If the thermal conductivities of nodes i and j are different, either because of different materials or variation with temperature, the conduction conductance is given by

$$U_{ij} = \frac{A_{ij}}{\frac{\Delta X_i}{k_i} + \frac{\Delta X_j}{k_j}}$$

Where ΔX_i = the conduction distance from the center of node i to the boundary between nodes i and j

ΔX_j = the conduction distance in node j

k_i, k_j = the thermal conductivity in nodes i and j respectively

Convection Conductors

The value of U_{ij} for covection between a fluid and a surface is given by

$$U_{ij} = hA$$

Where h = the convection coefficient

A = the convection area

Several methods are directly available in MOTAR for determining the heat transfer coefficient, h .

For flow in a tube the flow regime is assurred to be laminar when the Reynolds number is 2000 or less. For this regime the heat transfer coefficient is calculated by

$$h = \frac{k}{D}\left[3.66\ F1 + \frac{.0155\ F2}{\frac{1}{R_e P_r}\frac{X}{D} + .015\left[\frac{1}{R_e P_r}\frac{X}{D}\right]^{1/3}}\right] \qquad (16)$$

k = thermal conductivity

D = hydraulic diameter to flow

X = distance from tube enterance

Re = Reynolds number

$$= \frac{4\dot{m}}{\mu P}$$

$\dot{m}$ = flow rate of fluid

μ = viscosity of fluid

P = wetted perimeter of fluid flow passage

F1 = An input factor for modifying fully developed flow

F2 = An input factor for modifying developing flow

Equation (16) is a curve fit obtained by VMSC to approximate the Gratz solution to flow in a tube for values of $\frac{X}{D}\frac{1}{RePr}$ greater than 0.001.

The convection heat transfer coefficient for flow in a tube in the transition flow regime ($2000 < Re < 6400$) is approximated in MOTAR by the following relation:

$$h = \frac{K}{D}\left[0.116\,(Re^{2/3} - 125)(Pr)^{1/3}\right]$$

This relation was derived by Hausen and holds only for fully developed flow.

The relation used in MOTAR to determine h for turbulent flow ($Re \geq 6400$) is the following:

$$h = .023\,\frac{K}{D}\,(Re)^{.8}(Pr)^{1/3}$$

A more general option available on MOTAR for determining the heat transfer coefficient is given by the relation

$$St(PR)^{2/3} = f(Re)$$

Where St = Stantion number

$$= \frac{Nu}{Re\ Pr}$$

$$= \frac{h}{CpV}$$

V = Average fluid velocity

F(Re)= An arbitrary function of Reynonds number which the user can input as a table

The heat transfer coefficient is calculated by

$$h = \frac{K}{D}\,F(Re)\,Re(Pr)^{1/3}$$

Flow Conductors

As described in Section 3.1.1, equation (8) a flow conductor is needed to analyze the problem of a fluid flowing in a tube. The flow conductor is a one way conductor from j to i and is calculated by

$$UA_{ij} = \dot{W}\ Cp_i$$

Where

UA_{ij} = the conductance from the upstream lump

$\dot{W}$ = the mass flow rate in the tube

Cp_i = the fluid specific heat for lump i

The flow rate can be input directly or it may be obatined from a flow solution which is being performed simultaneously with the temperature solution problem (Section 3.2)

Radiation Conductors

The value of the condudance between two nodes, i and j, by radiation is given by

$$U_{ij} = \sigma\, \mathcal{F}\, A_{ij}\left([T_j+T_z]^2 + [T_i+T_z]^2\right)\left([T_j+T_z] + [T_i+T_z]\right)$$

Where σ = the Stefan-Boltzman Constant

$\mathcal{F}$ = the radiation interchange factor between nodes i and j

T_z = the value for conversion to absolute temperature

This relation is obtained by assuming that the heat transfer between nodes i and j by radiation is proportional to the temperature difference between these nodes during the iteration time span rather than proportional to the difference in the fourth power of the temperatures. This results in the above linearized coefficient. σ and T_z are input values so that the user may use any system of units for his problem. $\mathcal{F}A$ may be either input or calculated internally as described in Section 3.1.3.5.

3.1.3.2 Heat Exchanger Analysis

Four subroutines have been written to facilitate the thermal analysis of systems containing heat exchangers. These are HXCNT for analysis of counter flow heat exchangers, HXPAR for parallel flow exchangers, HXCROS for cross flow

exchangers and HXEFF for any heat exchanger with an input effectiveness. These subroutines calculate the outlet temperatures of two sides based upon the inlet temperatures and heat exchanger effectiveness. The relations used for effectiveness are given by the following equations taken from reference 17 for the first three subroutines.

Counterflow

$$\epsilon = \frac{1-e^{-\left[\frac{UA}{(MC)_s}\left\{1-\frac{(MC)_s}{(MC)_1}\right\}\right]}}{1-\frac{(MC)_s}{(MC)_1}\,e^{-\left[\frac{UA}{(MC)_s}\left\{1-\frac{(MC)_s}{(MC)_1}\right\}\right]}}$$

Where ϵ = effectiveness

UA = overall effectiveness

$(MC)_s$ = mass, specific heat product for the side with the smallest MC

$(MC)_l$ = mass, specific heat product for the side with the largest MC

The limiting case for this relation are:

(1) When $(MC)_s/(MC)_l = 0$,

$$\epsilon = 1 - e^{-UA/(MC)_s}$$

(2) When $(MC)_s/(MC)_l = 1$

$$\epsilon = \frac{\frac{UA}{(MC)_s}}{1+\frac{UA}{(MC)_s}} = \frac{UA}{(MC)_s + UA}$$

Parallel Flow

$$\epsilon = \frac{1-e^{-\frac{UA}{(MC)_s}\left[\frac{1+(MC)_s}{(MC)_l}\right]}}{1+\frac{(MC)_s}{(MC)_l}}$$

The limiting cases are

(1) When $(MC)_s/(MC)_l = 0$,

$$\epsilon = 1 - e^{-UA/(MC)_s}$$

(2) When $(MC)_S/(MC)_1 = 1.$,

$$\epsilon = \frac{1 - e^{-2\frac{UA}{(\overline{MC})_S}}}{2.0}$$

Cross Flow

A. Both Streams Unmixed

$$\epsilon = 1 - e^{\left[\left(e^{\left[-\frac{UA}{(\overline{MC})_S}\frac{(MC)_S}{(MC)_l}\eta\right]}-1\right)\frac{(MC)_l}{(MC)_S}\frac{1}{\eta}\right]}$$

Where $\eta = \left[\frac{(MC)_S}{UA}\right]^{0.22}$

B. Both Streams Mixed

$$\epsilon = \frac{\frac{UA}{(\overline{MC})_S}}{\frac{\frac{UA}{\overline{MC}_S}}{1-e^{-\frac{UA}{(\overline{MC})_S}}} + \frac{\frac{UA}{(\overline{MC})_l}}{1-e^{-\frac{UA}{(\overline{MC})_l}}}}$$

C. Stream $(MC)_S$ Unmixed

$$\epsilon = \frac{1 - e^{-\frac{(MC)_S}{(MC)_l}\left[1 - e^{-\frac{UA}{(\overline{MC})_S}}\right]}}{\frac{(MC)_S}{(MC)_l}}$$

D. Stream $(MC)_l$ Unmixed

$$\epsilon = 1 - e^{-\frac{(MC)_l}{(MC)_S}\left[1 - e^{-\frac{UA}{(\overline{MC})_l}}\right]}$$

Using the effectiveness as calculated by any of the above methods, the outlet temperatures are calculated as follows:

1. For the side with the smallest MC, $(MC)_s$:

$$Tout_s = Tin_s - \epsilon (Tin_s - Tin_l)$$

2. The outlet temperature for the side with the large MC is then calculated by

$$Tout_l = \frac{(MC)_s}{(MC)_l} (Tin_s - Tout_s) + Tin_l$$

3.1.3.3 Inline Heater Analysis

Provisons for the analysis of a fluid heater have been included in MOTAR with subroutine HEATER. This subroutine simulates an electrical heater with a control system which turns the heater on when a specified sensor lump drops below a set value and turns the heater off when the specified sensor lump rises above another set valve. When the heater is on an input quantity of heat is added to the heater node.

3.1.3.4 Cabin Analysis

A subroutine has been written for use with MOTAR which will give the user the ability to perform thermal and mass balance analyses on cabin air systems.

The cabin heat transfer and condensation analysis involves the two-component flow of a condensible vapor and a non-condensible gas, with condensation of the vapor occurring on surfaces in contact with the fluid. Two problems of this nature have been studied extensively.

1. Condensation on, or evaporation from, a surface over which a free stream of fluid is passing. In this case, for relatively low mass transfer rates, the fluid properties are assumed to be constant.

2. Dehumidification of a confined fluid stream by a bank of tubes. In this case there is a marked change in the temperature and vapor content of the fluid, and the detailed deposition of the condensate is not of primary interest. this type of analysis is usually handled on an overall basis similar to heat exchanges effectiveness calculations.

The following additional assumptions have been made with respect to the cabin atmospheric conditions.

1. The heat of circulation in the cabin is sufficiently high that the temperature and humidity are effectively the same throughout the cabin.

2. The velocity at all points where heat transfer and/or condensation can occur is known, and is propertional to the total mass flow rate in the cabin.

These assumptions make it possible to calculate the heat and vapor balance in the cabin for the entire volume as a unit, and to solve the heat transfer and condensation equations at each node independently of the other nodes.

Cabin humidity can be determined from an overall vapor balance in the cabin. The total vapor in the cabin at the end of an iteration is:

$$W_V = W_V^{\,i-1} + W_{V\ in} - W_{V\ out} - \Sigma W_L$$

Where
- W_V = mass of vapor in cabin at end of iteration i
- $W_V^{\,i-1}$ = mass of vapor in cabin at start of iteration i-1
- $W_{V\ in}$ = mass of vapor flowing into cabin during iteration i
- $W_{V\ out}$ = mass of vapor flowing out of cabin during iteration i
- ΣW_L = mass of vapor condensed during iteration i-1

$W_{V\ in}$ is determined from the known conditions of the gas flowing into the cabin.

$$W_{V\ in} = \dot{m}_{in} \left[\frac{\psi_{in}}{1.+\ \psi_{in}} \right]$$

Where
- $\dot{m}_{in}$ = mass flow rate into cabin
- ψ_{in} = specific humidity of gas flowing into cabin
- = time increment

It is assumed that an equal volume of gas is flowing out of the cabin. Then,

$$W_{V\ out} = \dot{m}_{out} \left[\frac{\psi_c}{1\ +\psi_c} \right]$$

Where
- ψ_c = specific humidity in the cabin (at the end of the previous iteration)

and $\dot{m}_{out} = \dot{m}_{in} [\rho_c / \rho_{in}]$

Where
- ρ_c = cabin density
- ρ_{in} = density of gas flowing into cabin

The condensation term ΣW_L is determined from the calculations for the individual nodes as described below. The properties of the cabin atmosphere are determined from the calculated

value of W_v. The vapor pressure in the cabin is

$$P_v = \frac{W_v}{V_c} R_v T_c$$

Where V_c = cabin volume
R_v = gas constant
T_c = temperature of cabin gas
P_v = vapor pressure

Assuming that the cabin pressure P_c is a constant, the gas partial pressure P_a is:

$$P_a = P_c - P_v$$

and $$W_a = \frac{P_a}{R_a T_c}$$

Where W_a = mass of non-condensible gas in the cabin.

Now the new value of specific humidity in the cabin can be determined by

$$\psi_c = \frac{W_v}{W_a}$$

The properties of the atmosphere can now be determined by

$$\mu_c = \frac{\chi\mu_g + \psi_c \mu_y}{\chi + \psi_c}$$

$$C_{pc} = \frac{C_{pg} + \psi_c C_{pv}}{1 + \psi_c}$$

$$k_c = \frac{\chi k_g + \psi_c k_y}{\chi + \psi_c}$$

$$\rho_c = \frac{W_v + W_s}{V_c}$$

Where μ = viscosity
C_p = specific heat
k = thermal conductivity
χ = molecular weight ratio, $\frac{M_v}{M_g}$

and all values are evaluated at T_c^{i-1}. Cabin temperature T_c can be determined by a heat balance on the cabin atmosphere.

$$T_c = T_c^{i-1} + \frac{\dot{m}_{in} C_{pc} (T_{in} - T_c^{i-1}) - \Sigma Q_L}{(W_v + W_A) C_{pc}}$$

Where T_c^{i-1} = T_c after previous iteration

T_{in} = temperature of gas flowing into cabin

ΣQ_L = net heat loss to cabin lumps

The heat transfer between the cabin atmosphere and the tube and structure lumps in the cabin is defined by:

$$Q_{Li} = hA_{Li} [Tc - T_{Li}] \Delta\tau$$

Where h = heat transfer coefficient

A_{Li} = heat transfer area of lump

T_{Li} = temperature of tube lump

$\Delta\tau$ = time increment

Using the Colburn-Chilton heat transfer-mass transfer analogy, the condensation (or evaporation) at the tube lump is determined by:

$$\Delta W_{Li} = K_m A_{Li} [P_v - P_{wi}] \Delta\tau$$

Where W_{Li} = condensation on wall, lb.

K_m = mass transfer coefficient

P_{wi} = vapor pressure at T_{Li}

The latent heat addition to the lump due to this condensation is

$$\Delta Q_\lambda = \Delta W_{Li} \lambda$$

Where λ = latent heat of vaporization (BTU/lb.)

The vapor pressure P_{wi} can be determined by a relationship derived from the Clausius-Clapeyron equation and the perfect gas law (Appendix K of Reference 16)

$$P_{wi} = P_o \; e^{\frac{\lambda}{R_g T_o}\left[\frac{T_{Li} - T_o}{T_{Li}}\right]}$$

Where P_o is known vapor pressure at a reference temperature T_o.

Three methods are available for determining mass and heat transfer coefficient. For tube lumps the equations from Reference 17 for gas flowing normal to the tube axis was assumed. Three different equations are used depending on the value of the Reynold's number.

$$Nu = 0.43 + .533\,(Re)^{.5}\,(Pr)^{.31} \quad Re < 4000$$

$$Nu = 0.43 + .193\,(Re)^{.618}\,(Pr)^{.31} \quad 4000 < Re < 40000$$

$$Nu = 0.43 + .0265\,(Re)^{.805}\,(Pr)^{.31} \quad 40000 < Re < 400000$$

These equations were derived for an air-vapor mixture, but should be relatively accurate for other similar gases. The Nusselt and Reynold's numbers in the equations are defined using the tube diameter for the characteristic dimension, and the velocity in the Reynold's number is input at each lump and ratioed to the total cabin atmosphere flow rate.

$$\nu i = \nu io \; \frac{\dot{W}c}{\dot{W}co}$$

Where $\dot{W}co$ = nominal cabin atmosphere circulation rate
νio = velocity at lump at $\dot{W}co$
$\dot{W}c$ = circulation rate at time of calculation

The second option assumes flat plate flow for cabin wall lumps. In this case the heat transfer coefficient, for laminar flow, varies along the plate. Hence, direction of gas flow and the location of an assumed leading edge must be assumed. The equation for flat plates from Reference is:

$$N_u = 0.332 \; Re^{.5} \; Pr^{1/3}$$

Where the Nusselt and Reynold's numbers are local values and are defined by the distance X from the assumed leading edge. For a wall lump of length L_j which is located a distance L_{jo} from the assumed leading edge, the average Nusselt number can be defined as:

$$N_u = 0.664 \ Pr^{1/3} \left[(Re_1)^{.5} - (Re_o)^{.5} \right]$$

Where N_u is defined by L_j

Re_o is defined by L_{jo}

Re_1 is defined by $L_{jo} + L_j$

The third option is a direct user input for convective heat transfer coefficient.

For the determination of mass transfer coefficients, the same equations as were used for heat transfer coefficient can be used with the Sherwood number substituted for Nusselt number and Schmidt number for Prandtl number. However, if the diffusion coefficient for the cabin is approximately equal to thermal diffusivity, the Sherwood number is equal to the Nusselt number and the mass transfer coefficient can be determined directly from the heat transfer coefficient. That is:

$$Sh = Nu$$

$$\frac{K_m R T_g x}{D} = \frac{h_x}{k}$$

If $D \simeq \alpha$ then

$$K_m = \frac{hD}{\alpha \rho C_p R T_g}$$

$$K_m \simeq \frac{h}{C_p P_c}$$

This is the Lewis relationship (Reference 17). For a mixture of oxygen and water vapor characteristic values are .866 for the diffusion coefficient, D, and .879 for thermal diffusivity, α, so the relationship should be valid.

For cabin tube and wall lumps the values for ΔQ_{Lj} and $\Delta Q_{\lambda i}$ are added to the basic heat balance equation for these lumps. Values for ΔQ_{Lj} are summed for all participating lumps for input to the cabin atmosphere heat balance. Values for $\Delta W L_j$ are also summed for all lumps for cabin humidiyt balance, and the value for total water condensed on each lump $W L_j$ is maintained.

If the rate of evaporation or condensation is high it would be possible for the cabin humidity to change significantly during a single iteration. This could lead, for example, to overestimating condensation by assuming that the humidity is constant in the calculation. A test of the approximate vapor pressure in the cabin at the end of the iteration is

made, and the condensation or evaporation at any lump is reduced, if the sign of the Δ W Li term is changed. A value W_v'' is calculated by:

$$W_v' = W_v^{i-1} - \Sigma W_L\, i$$

and

$$P_v' = \frac{W_v'}{144\ V_c} R_v\ T_g$$

Then for each lump if

$$\frac{P_v' - P_{wi}}{P_v - P_{wi}} < 0$$

a new value of $\Delta W_L i$ is calculated by:

$$\Delta W_{Li} = \Delta W_{Li} \left[\frac{P_v - P_{wi}}{P_v - P_v'}\right]$$

The new values of ΔW_{Li} are now again summed for the new value of $\Sigma \Delta W_L$ for establishing cabin humidity for the next iteration. A test is also made to assure that W_v' is never less than zero.

3.1.3.5 Radiation Interchange Analysis

Capabilities have been incorporated into MOTAR to facilitate the analysis of radiation heat transfer in an enclosure. The capabilities include the ability to:

(1) Analyze diffuse and/or specular infrared radiation in an enclosure
(2) Analyze diffuse and/or specular non-infrared radiation for as many wave bands as desired.
(3) Consolidate several temperature nodes into a single surface to improve computational efficiency

A radiation surface is defined as a group of temperature nodes which may be assumed to have identical radiating properties, angle factors and interchange factors.

The subroutines account for the net radiation heat transfer between a number of surfaces due to emitted radiation from each surface, reflected radiation from each surface, and radiation from any number of incident sources. The reflection of the energy originally emitted by another surface or from an external source may be either diffuse, specular, or any combination of the two.

Infrared Radiation

The radiosity of a surface is defined as the flux of infrared radiation leaving that surface with a diffuse distribution (according to Lambert's Law). That energy leaving a surface which has been reflected in a specular manner does not contribute to the radiosity of that surface. The incident infrared radiosity is denoted by the symbol H. The reflectance $(1-\epsilon)$ of a surface is separated into two components, the diffuse reflectance (ρ), and the specular reflectance (ρ^s). Here is the emmittance of the surface and is equivalent to the absorptance for long wavelength radiation. With the angle factors (Fij) defined in the normal way, there exist similar angle factors which relate the geometrical ability of surface i to radiate to surface j by means of a mirror-like reflection from specular surface k. Reference to Figure 1 indicates the method of imagery which will enable the calculation of these reflected angle factors. Here the angle factor to surface j is identical with the angle factor to the image of surface j. Also the angle factor is limited by the ability of surface i to "see" through the "window" of surface k. With the specular surface angle factors so defined, an interchange factor E_{ij} is defined similarly to reference 18 as follows:

$$E_{ij} = \sum_k \rho^s_k \; F_{ij\,(k)} + \sum_k \sum_l (\rho^s_k)(\rho^s_l) F_{ij(k,l)} + \cdots \quad (17)$$

Here $F_{ij(k)}$ is the angle factor from i to j as seen in the specular surface k, $F_{ij(k,l)}$ is the angle factor from i to j as seen in the double specular reflection from k and l. There are an infinite number of possible combinations of these multi-reflections. It is evident that the interchange factors account for the specularly reflected radiant flux from the reflecting surface. This portion of total leaving flux is not a component of the radiosity of that surface. The radiosity may be written

$$B_i = \epsilon_i \sigma T_i^4 + \rho_i H_i, \quad (18)$$

and, for ns surfaces,

$$H_i = \frac{1}{A_i} \sum_{j=1}^{ns} B_j A_j E_{ji} \quad (19)$$

Now the interchange factors obey the reciprocity relation

$$A_i E_{ij} = A_j E_{ji} \quad (20)$$

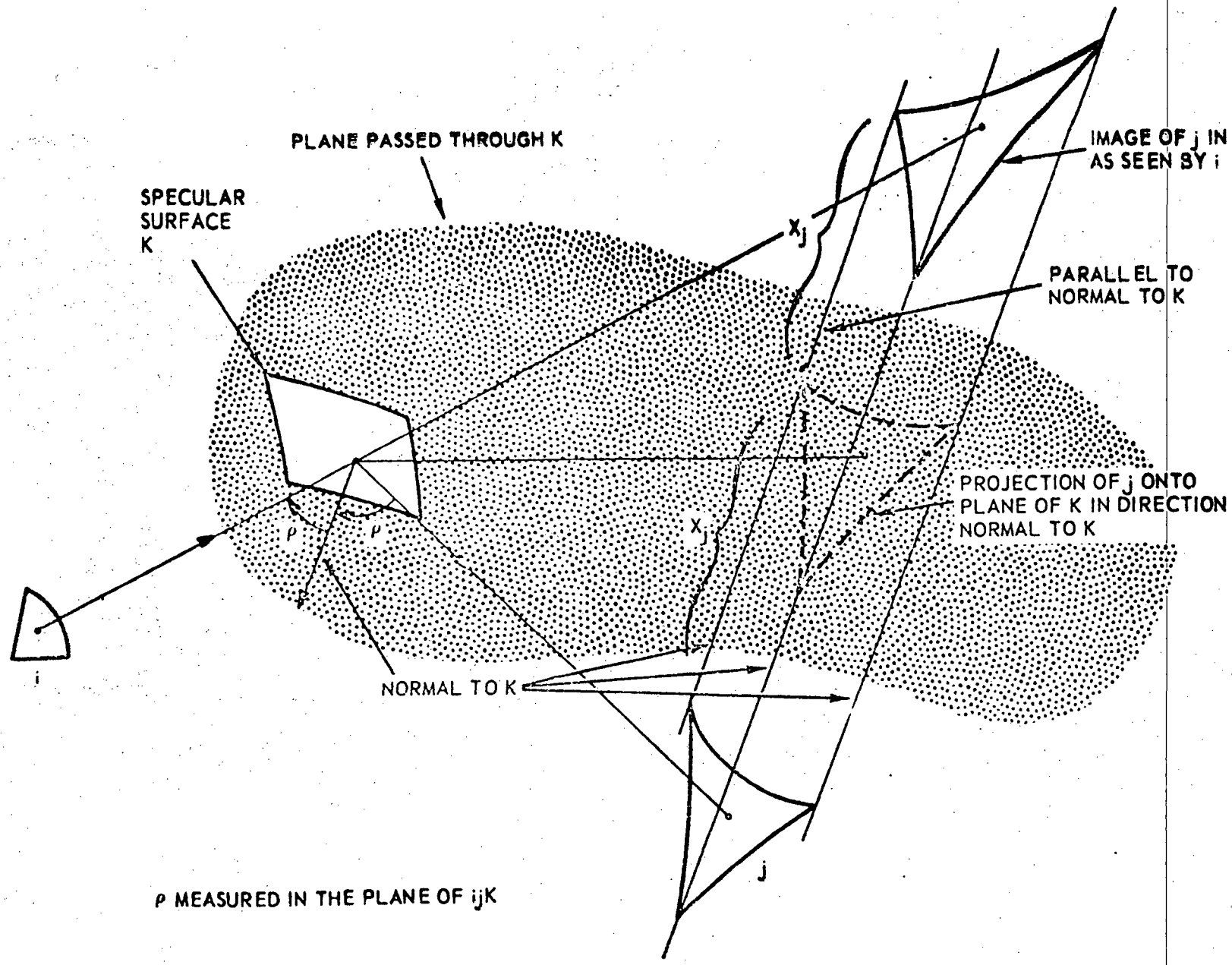


FIGURE 1 ILLUSTRATION OF METHOD USED TO DETERMINE SPECULAR SURFACE REFLECTED VIEW FACTORS

So,

$$H_i = \sum_j B_j E_{ij} \tag{21}$$

Substitution into the equation for B results in

$$\sum_j \left(\delta_{ij} - \rho_i E_{ij} \right) B_j = \epsilon_i \sigma T_i^4 \tag{22}$$

This equation represents a set of linear, simulteneous, inhomogeneous algebraic equations for the unknowns (B_j). The symbol δ_{ij} is the Kronecker delta function which is 1 when i = j and is 0 when $i \neq j$.

Note that the coefficients of B_j in equation (22) do not form a symetric coefficient matrix since the off diagonal terms contain $-\rho_i E_{ij}$. This equation can be made symetric by multiplying each equation by A_i / ρ_i. This gives

$$\sum_j \left(\frac{\delta_{ij} A_i}{\rho_i} - E_{ij} A_i \right) B_i = \frac{\epsilon_i A_i}{\rho_i} \sigma T^4 \qquad i = 1, ns \tag{23}$$

Written in matrix form this equation is

$$E\ B = T \tag{24}$$

Where E is a symetric coefficient matrix. The solution is

$$B = E^{-1} T = \left[e_{ij}^{-1} \right] T \tag{25}$$

or

$$B_i = \sum_{J=1}^{ns} e_{ij}^{-1} \frac{\epsilon i\ A_j}{\rho_j} \sigma T_j^4 \tag{26}$$

The net heat transfer rate asorbed by surface i is given by

$$Q_i = A_i \epsilon_i \left[H_i - \sigma T_i^4 \right] \tag{27}$$

Where H_i is given from equation

$$H_i = \frac{1}{\rho_i} \left[B_i - \epsilon \sigma T_i^4 \right]$$

Substituting in for H_i gives

$$Q_i = A_i \epsilon_i \left\{ \frac{1}{\rho_i} [B_i - \epsilon_i \sigma T_i^4] - \sigma T_i^4 \right\}$$

$$= \frac{A_i \epsilon_i}{\rho_i} \left\{ B_i - [\rho_i + \epsilon_i] \sigma T_i^4 \right\} \tag{28}$$

Substituting in for B_i from equation (26) into equation (28) gives

$$Q_i = \frac{A_i \epsilon_i}{\rho_i} \left\{ \sum_{J=1}^{ns} \frac{e_{ij}^{-1} \epsilon_j A_j}{\rho_j} \sigma T_j^4 - [\rho_i + \epsilon_j] \sigma T_i^4 \right\}$$

$$= \frac{A_i \epsilon_i}{\rho_i} \left\{ \sum_{\substack{J=1 \\ J \neq i}}^{ns} \frac{e_{ij}^{-1} \epsilon_j A_j}{\rho_j} \sigma T_j^4 - \left[\rho_i + \epsilon_i - \frac{e_{ij}^{-1} \epsilon_i A_i}{\rho_i} \right] \sigma T_i^4 \right\} \tag{29}$$

Since, in steady state, $Q_i = o$, and $T_i^4 = T_j^4$ for all i and j we can conclude that

$$\rho_i + \epsilon_i - \frac{e_{ij}^{-1} \epsilon_i A_i}{\rho_i} = \sum_{\substack{J=1 \\ J \neq i}}^{ns} e_{ij}^{-1} \frac{\epsilon_j A_j}{\rho_j}$$

Making the above substitution in equation (29) gives

$$Q_i = \sum_{J=1}^{ns} \sigma \frac{\epsilon_i \epsilon_j A_i A_j e_{ij}^{-1}}{\rho_i \rho_j} \left[T_j^4 - T_i^4 \right] \tag{30}$$

If we define $\mathcal{F}$ as

$$\mathcal{F}_{ij} = \frac{\epsilon_i \epsilon_j A_j e_{ij}^{-1}}{\rho_i \rho_j} \tag{31}$$

Then

$$Q_i = \sum_{J=1}^{ns} \sigma \mathcal{F}_{ij} A_i \left[T_j^4 - T_i^4 \right] \tag{32}$$

This equation gives the heat flux between surfaces. However, each surface can contain several nodes. The heat absorbed by for each node is determined by

$$Q_n = \frac{A_n}{A_i} \sum_{J=1}^{ns} \sigma \mathcal{F}_{ij} A_i \left[T_j^4 - T_n^4 \right] \tag{33}$$

Where n = the node number on surface i

Prior to each iteration, the temperature of the surfaces are determined by

$$T_i^4 = \frac{\sum_{n=1}^{nn} A_n T_n^4}{\sum_{n=1}^{nn} A_n} = \frac{\sum_{n=1}^{nn} A_n T_n^4}{A_i} \tag{34}$$

Where nn = the number of nodes on surface i

Since the heat transfer rate given by equation (33) depends on the node temperature, stability considerations must be taken into account. This is handled by storing the following relation into the array containing the sum of the conductors used for time increment calculation

$$CON_n = 4 \frac{A_n}{A_i} \sigma T_n^3 \sum_{J=1}^{nc} \mathcal{F} A_{ij} \tag{35}$$

Subroutine RADIR makes the calculations necessary to obtain Qn given by equation (33) and CONn given by equation (35). The following is a summary of the calculations:

A. The following are performed the first time through RADIR:

1. From the user input values of E_{ij}, A_i, and ρ_i, the E matrix given by equation (24) is formed. Only half of the symetric matrix is stored to save space.
2. The E matrix is inverted in its own space to get E^{-1} with elements e_{ij}
3. The $\mathcal{F}A_{ij}$ values are determined from equation (31) and stored in the surface connections data.

B. The following calculations are performed on each temperature iterations:

1. The temperature of each surface is calculated by equation (34).
2. The heat absorbed for each node is determined using equation (33) and is added to the Q array.

The routine utilizes data used for obtaining $\mathcal{F}A_{ij}$ in step A as working space for step B, thus, maximizing space utilization.

Radiation from External Source

As with the infrared radiation, the solar (or any other non-infrared radiation) interchange factor is defined by

$$E^*_{ij} = F_{ij} + \sum_k \rho^{*S}_k F_{ij}(k) + \sum_k \sum_l \rho^{*S}_k \rho^{*S}_l F_{ij}(k,l) + \cdots$$

Where ρ^{*S}_k is the solar specular reflectance of surface K

$F_{ij}(K)$ is the angle factor from i to j as seen in the specular surface

$F_{ij}(K,l)$ is the angle factor from i to j as seen in a double specular reflection from j to l to k back to i

The interchange factors as defined above accounts for the specularly flux reflected from the surface. Thus, since the specular component of the flux is assumed to go directly from surface i to surface j by the interchange factor, Eij, this portion of the total flux is not a component of the radiaty for the intermmediate surfaces (k and l above). The radiaty of surface i is given by

$$B^*_i = \rho^*_i H^*_i \tag{36}$$

Where B^*_i is the radocity (energy leaving)

H^*_i is the incident energy

ρ^*_i is the diffuse reflectance

The energy incident upon a surface is given by

$$H_i = \sum_{J=1}^{ns} B^*_j E^*_{ij} + S_i \tag{37}$$

Where S_i is the energy directly incident on surface i from an external source

Substituting equation (36) into (37), multiplying by A_i/ρ^*_i and simplifying gives the following relation for the radocity

$$\left[\frac{A_i}{\rho^*_i} - E^*_{ij}A_i\right] B^*_i + \sum_{\substack{J=1 \\ J\neq i}}^{n} E^*_{ij}A_i B^*_j = S_i A_i \qquad i=1,n \tag{38}$$

This set of n equations can be written in matrix form as

$$E^* B^* = S \tag{39}$$

Note that the equations are written so that E* is a symetric matrix, which has the solution for B*

$$B^* = E^{*-1} S \quad \text{or} \quad B_i = \sum_{J=1}^{n} [e^*_{ij}]^{-1} S_j \tag{40}$$

Where $[e^*_{ij}]^{-1}$ is the ijth element of the inverse of the E* matrix

The heat flux absorbed by the i th surface is given by

$$\frac{Q^*_i}{A_i} = \alpha H_i \tag{41}$$

But from equation (36)

$$H_i = \frac{B_i}{\rho^*_i} \tag{42}$$

Combining equations (40), (41), and (42) gives

$$Q^*_i = \sum_{J=1}^{n} e^{*-1}_{ij} \frac{\alpha_i}{\rho^*_i} A_j A_i S_j \tag{43}$$

If we define

$$\mathcal{F}^*_{ij} = e^{*-1}_{ij} \frac{\alpha_i}{\rho^*_i} A_j \tag{44}$$

Then the absorbed heat flux is given by

$$Q^*_i = \sum_{J=1}^{n} \mathcal{F}^*_{ij} A_i S_j \tag{45}$$

Equation (45) gives the heat absorbed by each surface. However, each surface may contain several temperature nodes. The absorbed heat for each node is given by

$$Q^*_n = \frac{A_n}{A_i} Q^*_i \tag{46}$$

Where A_n is the area of the node

Subroutine RADSØL was written to make necessary calculations to obtain Q_n^* given by equation (46). The following is a summary of the calculations:

A. The following calculations are made the first time through RADSOL:

1. From the user input values of E^*_{ij}, ρ_i^*, and Ai, the E* matrix given by equation (39) is formed. Only one half is stored since E* is symetric.
2. The E* matrix is inverted in its own space to get E^{*-1} with elements, e^{-1}_{ij}
3. The $\mathcal{F}^*_{ij}$ A_i values are determined from equation (44) and stored in the surface connections data.

B. The following calculations are performed on each temperature iteration:

1. The heat flux absorbed by each node is calculated by

$$\frac{Q_i^*}{A_i} = \frac{1}{A_i} \sum_{J=1}^{n} \mathcal{F}^*_{ij} A_i S_i$$

2. The net heat absorbed by this wave length radiation is calculated for each temperature node on each surface by

$$Q_n^* = A_n \frac{Q_i^*}{A_i}$$

This quantity of absorbed heat is added to the Q array for node n.

Note that the user may specify subroutine RADSOL for as many bands of non-infrared radiation is desired. A single call is required for each band.

3.2 PRESSURE - FLOW ANALYSIS

Provisions have been included in MOTAR for the pressured flow balance analysis of a flow system which contains a network of interconnected tubes including any combination of series/parallel flow. The effect of valves may be included in the pressure/flow analysis and the overall system pressure drop and flow rate may balanced with input pump pressure and flow characteristics.

3.2.1 Overall Model Description

The model used to mathematically describe a flow network consists of pressure "nodes" at tube junction and flow "conductors" for the tubes connecting the pressure nodes. The flow conductors for each tube is calculated as the reciprical of the sum of the flow resistance of temperature fluid lumps in each tube. The conductor/node network method for describing the flow system was chosen over that previously used because of the computational advantages and the flexibility it gives the user. The computational advantages come from the fact that the number of equations which must be

solved simultaneously are reduced from the number of tubes to the number of nodes between tubes. The method is more flexible because there are no restrictions on the number of tubes which can be connected at a junction.

For analysis purposes each flow network is divided into two levels. These are the "system" and any number of "subsystems". This subdivision is made to divide the network into those portions not requiring simultaneous equation solution (the system) and those requiring the solution to a set of equations(the subsystem). Also tied to the concepts of system and subsystem are the two basic classes of valves in MOTAR. These are (1) valves which dictate a given flow split based upon their position and (2) valves which dictate a resistance to flow based upon their position (class). Class 1 valves were devised for use in the system where no pressure balance is obtained and Class 2 valves were devised for use with the subsystem where the valve pressure drop is included in the pressure/flow balance.

Consider Figure 2 to help illustrate the meaning of system and subsystem. In Figure 2 (a) if value VI is of the specified flow split type (class 1) then, the system consists of tubes 1, 2, 11, 12,18, and 19. This is because the flow in each of these tubes is defined by the flow in tube 1 (the inlet) and the value position. Two subsystems exist for this example which are those requiring a pressure balance. One consists of tubes 3 thru 10; the other consists of tubes 13 thru 17. In this example, if valve VI were of the pressure drop type (class 2) then the system would be tubes 1 and 19 and one subsystem would exist consisting of tubes 2 thru 18.

Figure 2(b) shows an example in which no subsystem is required. If the valves V1 and V2 are both Class 1 (flow splitting) then all tubes are in the system and no subsystem exists. If valve V1 is class 2, then valve V2 must also be Class 2 and tubes 2 thru 6 belong to the subsystem. If V1 is Class 1 and V2 is Class 2, then all tubes are in the system except 3 and 4 which are in the subsystem.

The sections that follow give a detailed description of the analytical methods for (1) determining the conductor values for each tube, (2) determining the conductor values for valves, (3) determining network solutions, and (4) balancing system pressure drop and flow with pump flow characteristics.

3.2.2 Tube Conductor Determination

The value of the flow conductor is determined for each tube by first calculating the flow resistance for each temperature fluid lump contained in the tube, summing these resistances up to obtain the flow resistance of the tube and inverting the tube resistance to get the conductance Flow conductance is defined by the relationship

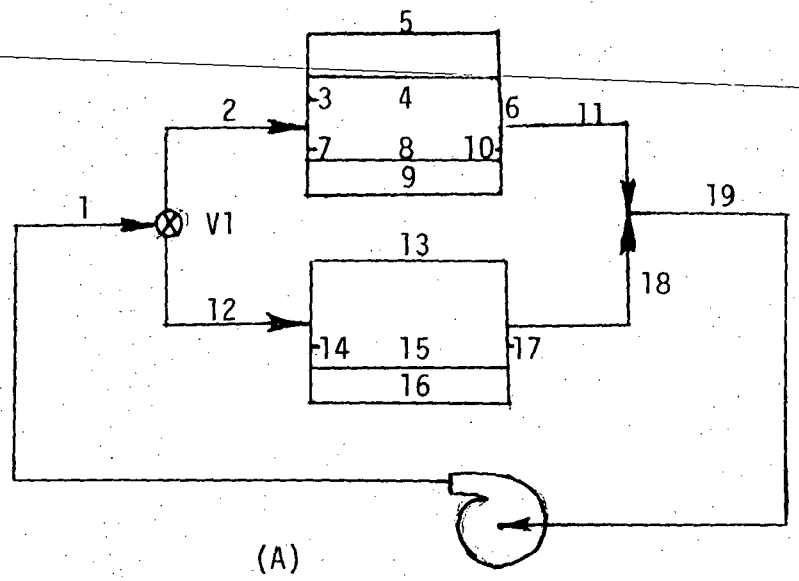
$$\dot{W}_{ij} = G_{ij} [P_i - P_j] \qquad (47)$$

Where W_{ij} = flow rate between pressure nodes i and j

G_{ij} = flow conductance between nodes i and j

P_i = pressure at pressure node i

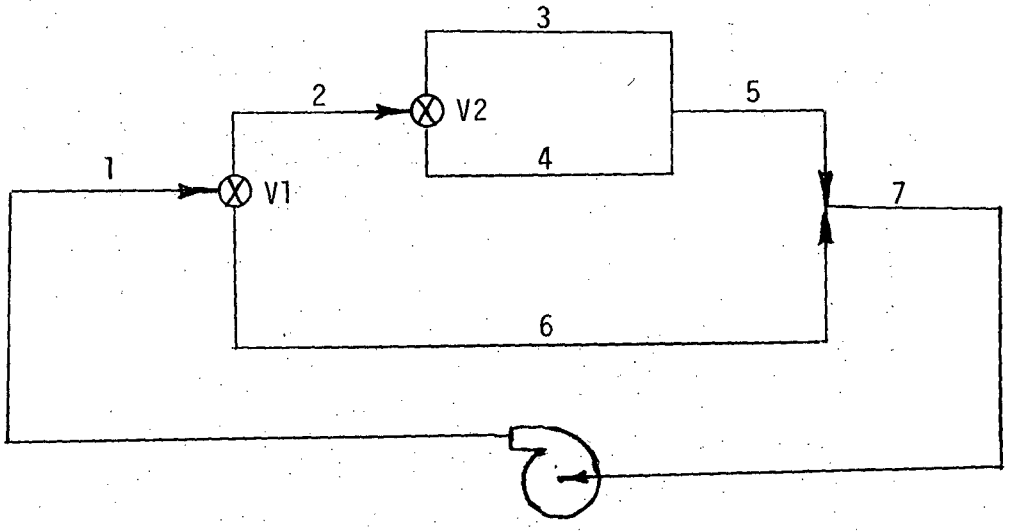
P_j = pressure at pressure node j



(A)

Numbers are tube numbers

V1 = Valve No. 1



(B)

Figure 2: Illustration of System and Subsystem Concept

The flow resistance for each lump is then

$$R_K = \frac{1}{G} = \frac{\Delta P_k}{\dot{W}}$$

Where R_k = flow resistance for lump k

ΔP_k = pressure drop for lump k

But ΔP_k is given by

$$\Delta P_k = f_k \; ffc \; \frac{L_k}{D_k} + \quad k \quad \frac{W^2}{2g_c \rho_k A^2} \tag{48}$$

Where f_k = the friction factor for lump k

ffc = the friction factor coefficient

L_k = the lump length for lump k

D = the lump hydraulic diameter for lump k

$_k$ = the dynamic head losses for lump k

$\dot{W}$ = the flow rate

g_c = the gravitational constant

ρ_k = the fluid density for lump k

A = the flow area

The flow resistance is then given by

$$R_k = f_k \; ffc \; \frac{L_k}{D_k} + \quad k \quad \frac{\dot{W}}{2g_c \rho_k A^2} \tag{49}$$

Two options are available for obtaining the friction factor, f_k. These are (1) it is calculated internally and (2) it is calculated internally for laminar flow but is obtained from a table of f vs Re (where Re is the Reynold's number) for transition and turbulent flow. For the first option the internal calculations for the three flow regimes are:

Laminar Regime: $Re_k \leq 2000.$

$$f_k = \frac{64}{Re_k} \tag{50}$$

Where f_k = friction factor for lump k

Re_k = Reynolds number for lump k

Transition Regime: $2000 < Re_k < 4000$

$$f_k = .2086082052 - .1868265324\left[\frac{Re_k}{1000}\right] + .06236703785\left[\frac{Re_k}{1000}\right]^2 - .0065545818\left[\frac{Re_k}{1000}\right]^3 \quad (51)$$

Turbulent Regime: $Re_k \geq 4000$

$$f_k = \frac{.316}{Re_k^{.25}} \quad (52)$$

Equation (51) for the transition regime is a curve fit between the laminar and turbulent regimes which was derived to match the two curves in a continuous manner. It is merely an arbitrary curve in this undefined region. A curve of the friction factor VS Reynold's number given by the above relations is shown in Figure 3.

The second option for friction factor uses equation (50) for the laminar regime and a user input curve of f_k vs Re for the other regimes. The options available for input of the dynamic head loss, $\mathcal{K}$, include (1) an input constant or (2) a tabulated curve of $\mathcal{K}$ vs Re.

To obtain the conductance for each tube, the flow resistances for all the lumps in the tube are added and then inverted. That is,

$$G_{ij} = \frac{1}{\sum_k R_k} \quad (53)$$

3.2.3 Valve Analysis

As discussed in Section 3.2.1 two classes of valves are available in MOTAR. For Class 1 valves the valve position dictates directly the friction of the incomming flow distributed to the valve outlet branches. For Class 2 valves the valve position and pressure drop characteristics define the flow resistance for the valve branches. These resistances are added to the other flow resistance of the tubes to obtain the over all tube conductance prior to solving for the flow rate.

A number of valve types are available to the user for both classes of valves which are:

(1) Rate limited
(2) Polynomial (replaces the polynomial)
(3) Shut-off propoitioning

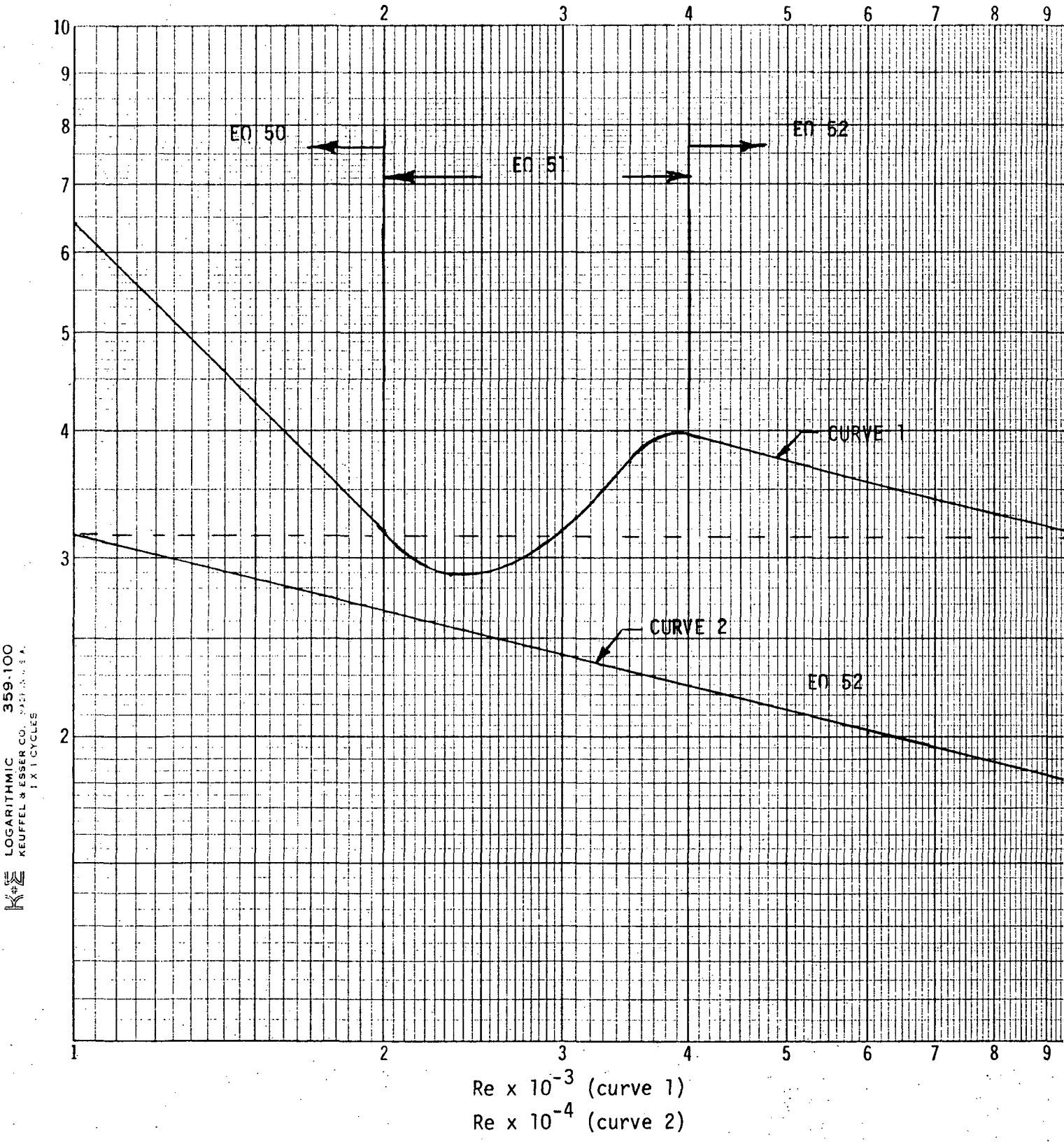


FIGURE 3 Friction Factor vs Reynolds Number

These valves types differ in the relationship between the sensor and the valve position as described below.

3.2.3.1 Valve Position Determination

Described below are the methods used in the routine to obtain the valve position. The following section will describe the use of the valve position to obtain flow split and pressure drop information.

Rate Limited Valve

The valve position for the rate limited valve is obtained by an approximate integration of the valve rate of movement, $\dot{X}$. $\dot{X}$ depends on the temperature difference between the valve control set point temperature and the sensor temperature as shown in Figure 4. With this characteristic, the valve has no movement as long as the valve temperature error, ΔT, is within the dead band. Outside the dead band, the velocity of the valve increases linerarly as the error increases to a maximum rate, $\dot{X}$ max. The dead band, rate of velocity increase, $d\dot{X}/d(\Delta T)$, and the maximum velocity are controlled by user input.

The relations used to obtain the valve positions are as follows:

$$X^{i+1} = X^{i} + (\dot{X}^{i+1})\ (\Delta\tau) \qquad (54)$$

Where X^{i+1} = valve position at iteration i+1
X^{i} = valve position at iteration i
$\dot{X}^{i+1}$ = valve velocity at iteration i+1
$\Delta\tau$ = the problem time increment

The valve position is limited by

$$X\ \text{min} \leq X^{i+1} \leq X\ \text{max}$$

Where X min and X max are input limits on the valve position. The valve velocity, $\dot{X}^{x+1}$, in equation (54) is given by:

$$\dot{X}^{i+1} = 0 \text{ if } \left|Tsen - Tset\right| \leq Tdb$$

Where

Tsen = Sensor lump temperature

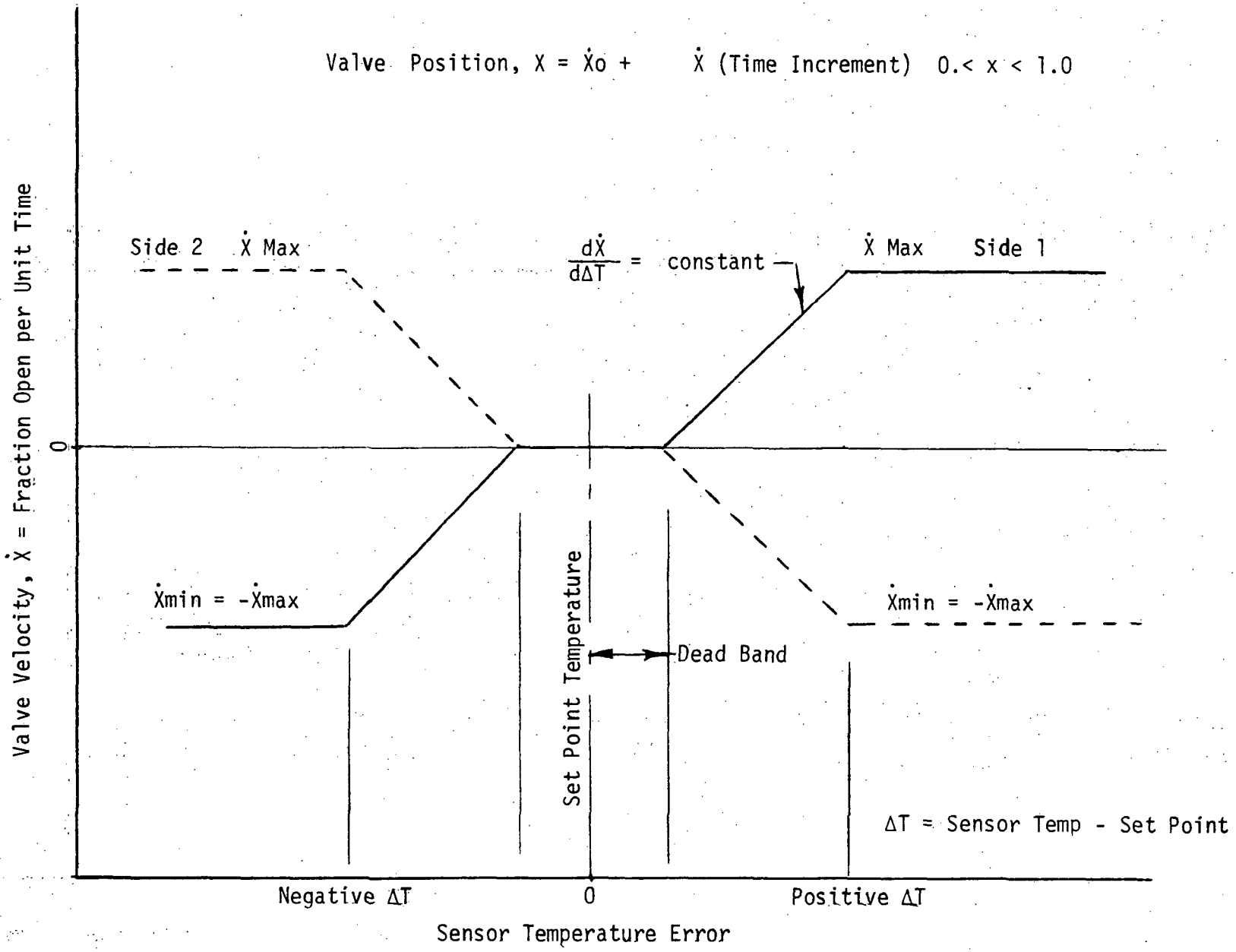


Figure 4 Rate Limited Valve Operation

Tset = Set point temperature

Tdb = Valve dead band temperature

$$\dot{X}^{i+1} = \frac{d\dot{X}}{d(\Delta T)} \quad [\ Tsen-Tset-Tdb\ ] \text{ if } Tsen > Tset + Tdb$$

$$\dot{X}^{i+1} = \frac{d\dot{X}}{d(\Delta T)} \quad [\ Tsen-Tset+Tdb\ ] \text{ if } Tsen < Tset - Tdb$$

The valve velocity is limited by

$$\dot{X}min \leq \dot{X}^{i+1} \leq \dot{X}max$$

Polynomial Valve

The polynomial valve determines the steady state valve position as a forth degree polynomial function of the temperature error between the sensor lump and the set point. A valve time constant is then applied to determine how far between the previous position and the new steady state position the valve will move. The steady state position, X_{ss}, is given by

$$X_{ss} = Ao + A_1 \Delta T + A_2 \Delta T^2 + A_3 \Delta T^3 + A_4 \Delta T^4$$

Where ΔT = Tsen - Tset

Tsen = the sensor lump temperature

Tset = the set point temperature

Ao, A_1, A_2, A_3, A_4 = input constants

The valve position, X^{i+1} is then determined by

$$X^{i+1} = X_{ss} + (X^i - X_{ss})\, e^{-\Delta r / \tau_c}$$

Where X^{i+1} = valve positon at iteration i+1

X^i = valve position at iteration i

Δr = problem time increment

τ_c = valve time constant

Note that this valve combines the capabilities of the polynomial valve and the proportioning valve described in Reference 12. If one desires to eliminate the effect of the time constant (and thus, give the valve an instantaneous response), a valve for r_c should be input which is small compared to the time increment, Δr . Also, either a constant value or a temperatu lump number may be specified for the set point to permit the use of the value for proportioning between two sides.

Shut-off Valve

Two types of shut-off valves are available. For the first, the valve position decreases from Xmax to Xmin when the temperature of the sensor lump drops below the specified "off" temperature Toff and increased from Xmin to Xmax when the sensor lump exceeds a second specified temperature, Ton. Ton must be greater than Toff. The second type of shut-off valve works in reverse to the first. The valve position increases from Xmin to Xmax when the sensor temperature drops below the specified Ton and decreases from Xmax to Xmin when the sensor lump increases above the off temperature, Toff. For the second type, Toff must be greater than Ton.

3.2.3.2 Flow Split and Pressure Characteristic Determination

For the Class 1 valves the valve position as determined by the methods described in Section 3.2.3.1 will determine the flow split directly between two outlet tubes. (Two are required for class 1). The flow rate is given by

$$\dot{W}_1 = X\,\dot{W}in$$
$$\dot{W}_2 = (1.-X)\,\dot{W}in \qquad (55)$$

Where X = the valve fraction of the total travel from side 1
$\dot{W}in$ = flow rate into the valve
$\dot{W}_1$ = flow rate out side 1
W_2 = flow rate out side 2

For the Class 2 valves, the position is used to determine the valve resistance to flow. The valve pressure drop on one side is assumed to be

given by

$$\Delta P = E \left[\frac{\dot{W}}{X} \right]^2 \tag{56}$$

Where E is an input constant

W is the flow through one side of the valve

X is the valve position (fraction of total possible distance)

Since flow resistance is $\Delta P/\dot{W}$, the valve flow resistance is given by

$$R_v = \frac{E \dot{W}}{X^2} \tag{57}$$

This value of flow resistance is calculated and added to the other flow resistances of the tube prior to performing the operation in equation (53) to find the tube conductor.

Class 2 valves may be either one way or two way - i.e., there may be one tube or two tubes at the outlet. If only one tube exists on the valve outlet the flow resistance is calculated using equation (57) above. If a second tube exists, the resistance on side 2 is given by

$$R_{v2} = \frac{E_2 \dot{W}_2}{(1 - X)^2} \tag{58}$$

3.2.4 Pressure-Flow Network Solution

After the flow conductor values have been obtained by the methods described in Sections 3.2.2 and 3.2.3 a set of simultaneous equations are set up and solved for each subsystem. This set of equations are obtained by conservation of mass at each pressure node. This gives as many equations as there are nodes in the subsystem.

For any node i the conservation equation can be written as follows.

$$\sum W_{out} - \sum W_{in} = 0 \tag{59}$$

Let $W_{in} = W_i$

and $$\sum W_{out} = \sum_{J=1}^{nc} G_{ij} [P_j - P_i] \tag{60}$$

Then equation (59) becomes

$$\sum_{J=1}^{n} G_{ij} [P_j - P_i] - W_i = 0 \qquad i=1,n \tag{60}$$

Where G_{ij} = flow conductor between pressure nodes i and j
P_i = pressure at node i
P_j = pressure at node j
W_i = flow rate added at node i
N = number of pressure nodes in the subsystem

The above equation can be written as a set of simultaneous equations in P and solved for all pressures. One pressure in the system must be specified. The set of equation can be written in Matrix from (assuming P_n is the specified pressure):

$$GP = C \tag{61}$$

Where

$$G = \begin{bmatrix} \sum G_{1j} & -G_{12} & -G_{13} \cdots\cdots\cdots \\ -G_{21} & \sum G_{2j} & -G_{23} \quad \cdots\cdots\cdots \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -G_{n-1,1} & -G_{n-1,2} & \text{--------} \sum G_{n-1,j} \end{bmatrix}$$

$$P = \begin{bmatrix} P_1 \\ P_2 \\ \cdot \\ \cdot \\ \cdot \\ P_{n-1} \end{bmatrix}$$

$$C = \begin{bmatrix} W_1 + G_{1n} P_n \\ W_2 + G_{2n} P_n \\ \cdot \\ \cdot \\ \cdot \\ W_{n-1} + G_{n-1} P_n \end{bmatrix}$$

Specified Pressure (Doesn't have to be the last one)

The above equations are solved for pressures at each point in the system and flow rates are then calculated for each tube (conductor) by:

$$\dot{W}_{ij} = G_{ij} (P_i - P_j) \qquad (62)$$

Two methods of solution are available in MOTAR for solving the set of equations given by equation (61). These are the Gauss-Jordon reduction method, which is a direct solution method and the Seidel stationary point Iteration method with successive overrelaxation which is an iterative method. The user may specify which of the two methods to be used. Normally, for the typical problem the direct method would be used. However, for extremely large problems where the direct method proves unsatisfactory, the iterative method may be an improvement.

Since the flow conductors are functions of the flow rate, the set of equations given by (61) are solved numerous times on each temperature iteration with a new set of G_{ij} values for each solution. The iteration process continues until the change in the flow rates is within some user specified tolerance before proceeding to the next iteration.

3.2.5 Pump and System Pressure - Flow Matching

Concurrent with iterating the system flow equation to solution on each iteration, the overall system pressure drop and flow rate must be matched to a pump characteristic. Several types of pump characteristics

are available to the user as options. These are (1) The system flow rate may be specified as a constant, (2) it may be specified as a known function of time, (3) the pressure drop may be specified as a function of the flow rate in a tabular form and (4) the pressure drop may be specified as a function of flow rate with a fourth degree polynomial curve.

The first two options require no balancing of the pump with the system. Balancing is required for options (3) and (4) and iterative procedures have been devised to obtain the solution of the pump curve to the system characteristics with as few passes as possible through the system pressure/flow balancing loop for these options. The procedures used for these options are described below.

3.2.5.1 Tabular Pump Curve Solution

The matching of a tabulated pump pressure rise/flow characteristic to the system pressure drop/flow characteristic is accomplished by the following precedure. See Figure 5 to aid in following the procedure.

Step 1: The initial flow rate, W_1, at the system inlet is established either from user input on the first iteration or the system flow of the previous iteration for subsequent iterations.

Step 2: Using $\dot{W}_1$, a solution to the flow network is obtained using the methods described in sections 3.2.2, 3.2.3 and 3.2.4. Following this solution, ΔP_1 is available establishing point 1 on the true system characteristic curve shown in Figure 5.

Step 3: The constants which describe the straight line approximation for the system pressure/flow characteristic is established. (line 0 to 1 for the first pass, line 1 to 2 for the second pass, etc.)

$$\Delta P_s = C \dot{W}_s + D$$

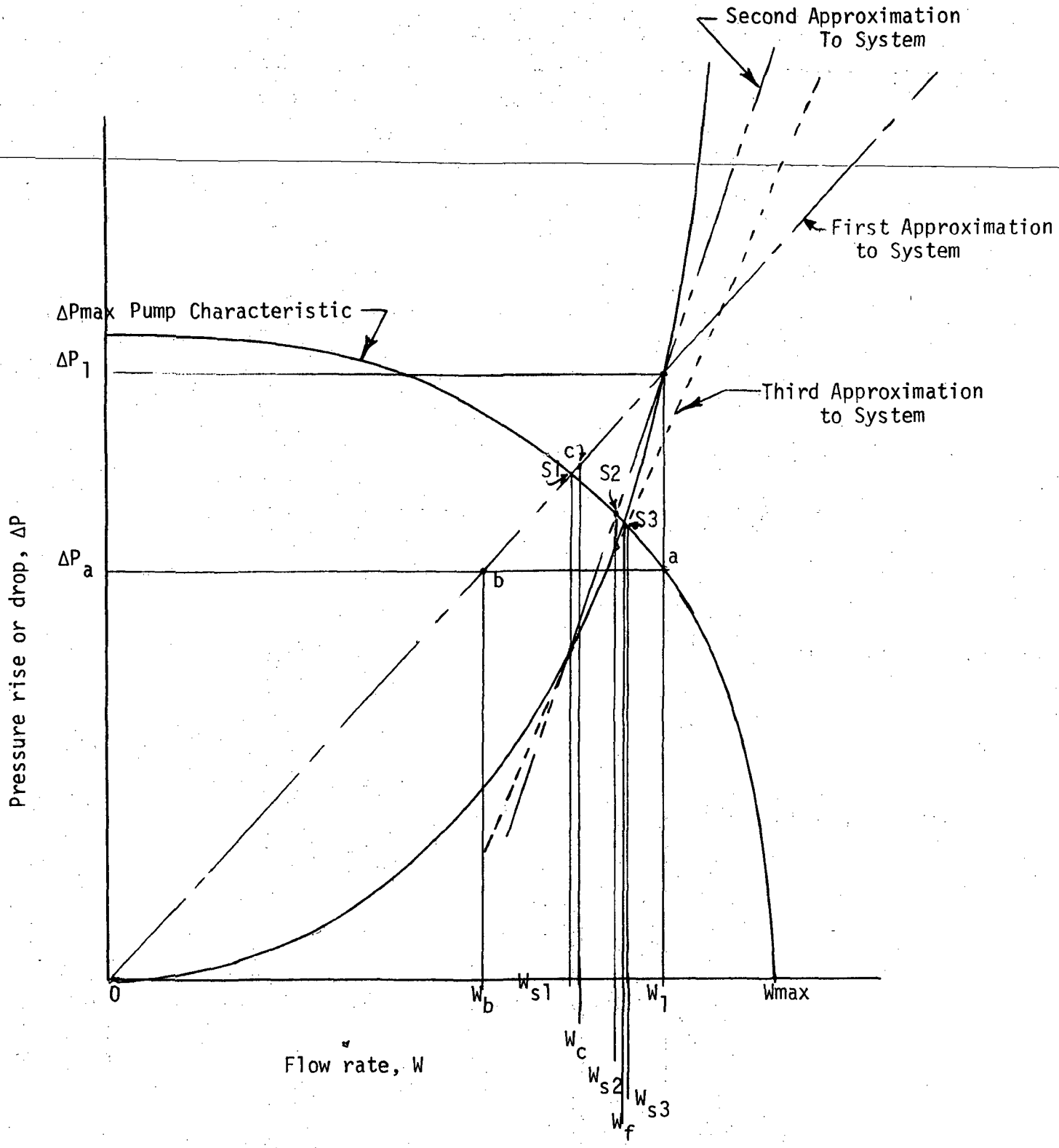


Figure 5 System/Pump Curve Solution

(a) For the first pass,

$$C = \frac{\Delta P_1}{W_1}$$

$$D = 0$$

(b) For the second and subsequent pass on a given iteration,

$$C = \frac{\Delta P_2 - \Delta P_1}{W_s - W_1}$$

$$D = \Delta P_1 - \left[\frac{\Delta P_2 - \Delta P_1}{W_s - W_1}\right] W_1$$

Where ΔP_2 and W_s are the values for the last solution on the system characteristic (point 2 on Figure 5 for the third pass)

ΔP_1 and W_1 are the values for the next to the last solution on the system characteristic (point 1 for the third pass)

Step 4: The solution to the approximate system characteristic and the tabulated pump characteristic is determined by the following iterative procedure

(a) Start with Wa = $\dot{W}1$

(b) Determine ΔP_a by interpolating the pump curve at W_a

(c) Determine the flow rate given by the approximate system characteristic at ΔPa, $\dot{W}b$

$$Wb = \frac{\Delta Pa - D}{C}$$

Where constants C and D were obtained in Step 3.

(d) Determine the approximate solution, W_c by

$$W_c = \frac{Wa + Wb}{2}$$

(e) Check the tolerence:

$$\text{Is} \quad (W_c - W_a) \quad < \quad .001\ W_a$$

(f) If the inequality does not hold, set $W_a = W_c$ and repeat b thru f. If the inequality does hold W_c is the solution between the pump curve and the latest approximate to the system characteristic and

$$W_{si} = W_c$$

Step 5: Check the following tolerence

$$\left| W_1 - W_{s1} \right| = E$$

If $E < .001 * W_1$, then W_{s1} and ΔP, are the solution. If the tolerance is not met, repeat steps 2 thru 5 using the latest flow rate, W_{s1} as W_1 and find W_{s2}, W_{s3} etc. until W_{sf}, the final solution is located.

3.2.5.2 Polynomial Pump Curve Solution

When the user describes the pump curve with a polynomial curve fit, the pump characteristic is described by the relation

$$\Delta P_p = Ao + A_1 W + A_2 W^2 + A_3 W^3 + A_4 W^4$$

When this option is used, the procedure for matching the pump characteristic described above to the system characteristic is identical to that described in Section 3.2.5.1 for the tabulated pump characteristic except Step 4 is replaced with the following Step 4A:

Step 4A: Set up the equation for $\dot{W}_s$ and solve.

(a) Set up the equation

Since:

$$\Delta P_p - \Delta P_s = 0$$

$$\Delta P_s = C\, W_s + D$$ (C and D are obtained from Step 3, equation (64 or 65)

$$\Delta P_p = Ao + A_1 W_p + A_2 W_p^2 + A_3 W_p^3 + A_4 W_p^4$$

$$W_s = W_p$$

Then the equation for $\dot{W}_s$ is

$$(Ao-D) + (A_1-C)\, W_s + A_2 W_s^2 + A_3 W_s^3 + A_4 W_s^4 = 0$$

(b) Solve the equation for W_s using the Newton-Raphson Method of solution for a fourth order polynomial

The remaining steps are identical to that given in Section 3.2.5.1.

4.0 ROUTINE OPERATIONAL DESCRIPTION

This section describes the nature of the MOTAR computer routine. The operation of MOTAR is divisible into three phases. These are: (1) the preprocessing phase which includes the reading of input data, the assembling of computer generated subroutines, and processing the input data into a compressed data tape, (2) the compilation phase in which the computer generated and user assembled subroutine are compiled and (3) the processing phase during which the compressed data tape is read and the desired computation are made. A schematic of the overall flow of the MOTAR routine is given in Figure 6 . This three phase procedure was devised (1) to permit the tailoring of the main temperature calculation subroutines to conform with the requirements of the data in the most efficient manner and (2) to permit the user to perform logical operations and call user subroutines in the input data.

A description of the three phases including a summary of the subroutines used in each phase is given in the following subsections. A listing of all subroutines is provided in Appendix D .

4.1 PREPROCESSING PHASE

During the preprocessing phase of MOTAR the following tasks are performed:

(1) A data tape is generated and/or edited when the user so requests.

(2) The input data, which is input in a free field format, is read from cards or the final data tape,processed for more efficient use by the computer and stored on a compressed data tape.

(3) The user logic supplied in the $PRETEMP, $POSTTEMP, $CENTRAL, and $OUTPUT is read and written in a form compatible with the Fortian compiler.

(4) The symbolic logic for the transient and steady state temperature calculation subroutine is generated in a form compatible with the FORTRAN compiler.

A brief summary of the elements used during the preprocessing phase is given below:

STEP1 — Specifies the overlay structure of the preprocessing phase. Elements NTWRK, SUBFLW, SUBCRV, OPBLOK, and GETCOM overlayed. MAIN is designated as the Main preprocessing phase routine.

MAIN — The driving routine for the preprocessing phase. It calls on subroutines SUBA and SUBB to generate a data tape, read input and write a compressed

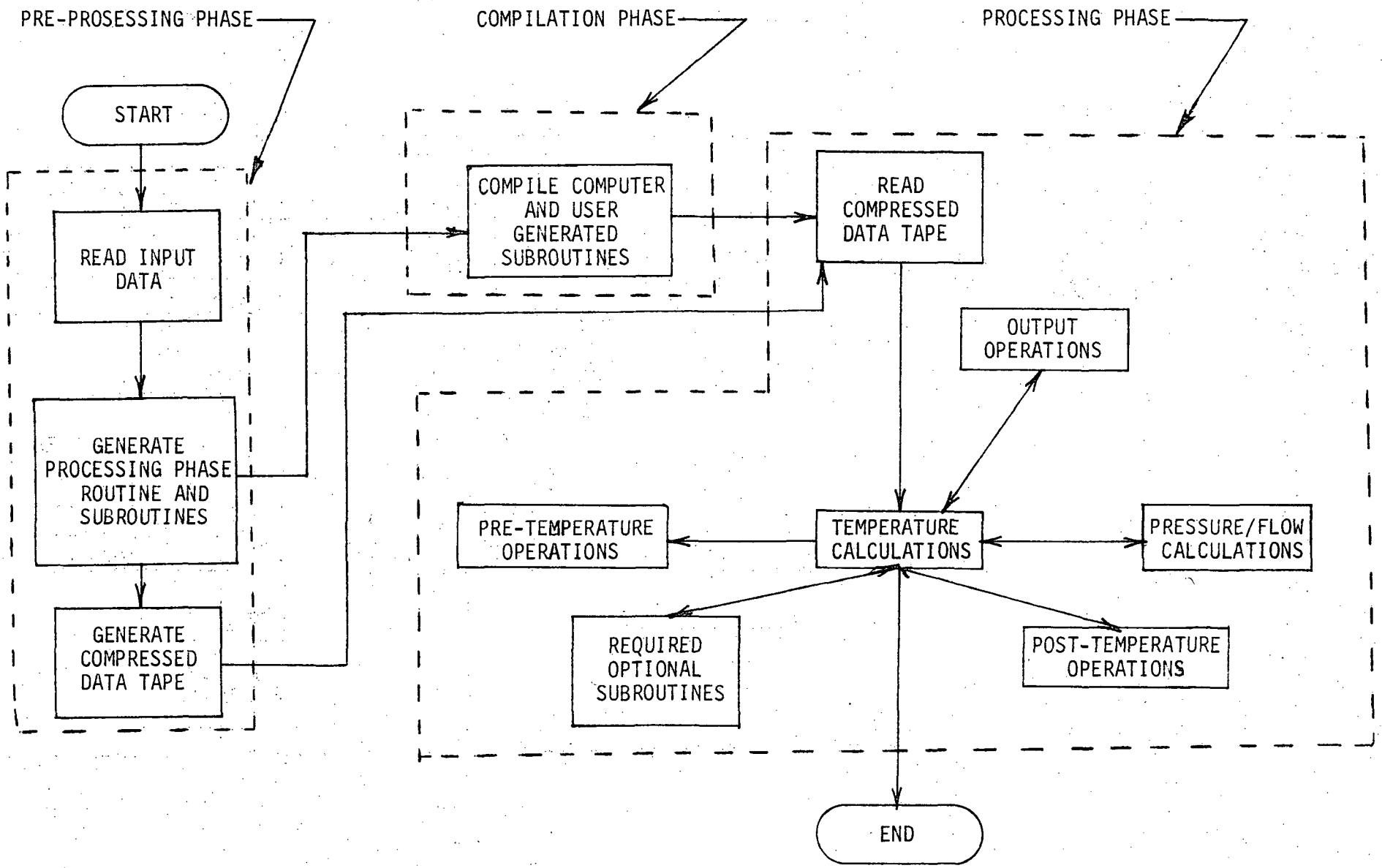


FIGURE 6: MOTAR ORGANIZATION

data tape, read user logic and write it in required symbolic form and generate temperature solution subroutines corresponding to the requirements of the input data. They call on the following subroutines to perform this: EDIT, COLCHK, NTWRK, SUBFLW, SUBCRV, PRETMP, CENTRL, PSTTMP, SUBOUT, ISUBA, KRVS, GENOUT, GETCON, BLOCK. These are described below.

A) EDIT — Places the user input data on a magnetic tape and performs edits to previously generated magnetic tapes. Calls EXIT if an error occurs.

B) COLCHK — Reads and writes data cards, skips blank columns and locates delimiters.

C) NTWRK — Reads and processes the $NETWORK data. Calls the following subroutines to handle different types of data:

(1) COLCHK - (Described above)

(2) INTMP - Reads and processes INITIAL TEMPERATURE data

(3) CAPHT - Reads and processes +CAPACITANCE and + ABSORBED HEAT data

(4) SUBCON - Reads and processes 'CONDUCTION' conductor data

(5) SUBRAD - Reads and processes 'RADIATION' conductor data

(6) SUBCNV - Reads and processes 'CONVECTION' conductor data

D) SUBFLW — Reads and processes $FLOW SYSTEMS data

E) SUBCRV — Reads $CURVES data and writes it on a drum for subsequent processing

F) CENTRAL — (Entry point to OPBLOK) Reads user logic in $CENTRAL block and writes it on a drum for subsequent processing

G) PREOP — (Entry point to OPBLOK) Reads user logic in $PRETEMP block and writes it on a drum for subsequent processing

H) POSTOP (Entry point to OPBLOK) Reads user logic in $POSTTEMP block and writes it on a drum for subsequent processing

I) SUBOUT (Entry point to OPBLOK) Reads user logic in $OUTPUT block and writes it on a drum for subsequent processing

J) KRVS Process $CURVES data which was read by subroutine SUBCRV

K) GETCON Rearranges conductor connections data for the implicit temperature routines when required

L) BLOCK Writes computer generated and user logic elements in a form expected by the Fortran compiler

Other subroutines called during the preprocessing phase are described below:

EXIT	Terminates the job
NODENO	Reads in node numbers for all MOTAR options including multiple input options
SUBI	Reads in integers
SUBF	Reads in real numbers
SUBS	Converts arguments from a field data character to an integer by means of an Alpha-numeric search
ORDER	Orders blocks of data in an array so that one element of the block is in either ascending or decending order from block to block.
CONDNO	Reads in conductor numbers and connected nodes including Multiple input options
GENOUT	List an array of mixed mode numbers
GENR	(Entry point to GENOUT) Lists an array of real numbers
GENI	(Entry point ot GENOUT) Lists an array of integers

4.2 COMPILATION PHASE

During the compilation phase, a compilation is performed on as many as six subroutines which were setup during the preprocessing phase and any other subroutines which the user may desire to supply. The six subroutines which will normally require compilation are tabulated below:

STEP2	The main routine for the processing phase. Contains the $CENTRAL user logic as well as logic for reading and writing a dump tape
PRETMP	A subroutine containing the logic from the $PRETEMP
POSTMP	A subroutine containing the user logic from the $POSTTEMP block
OUTPUT	A subroutine containing the user logic from the $OUTPUT block
EXPLCT or IMPLCT	The transient temperature solution subroutine which is generated during the preprocessing phase based upon the requirements of the input data
EXPSS or IMPSS	The steady state temperature solution subroutine which is generated during the preprocessing phase based upon the requirements of the input data

The user must also compile any other subroutines required for the problem during the compilation phase.

4.3 PROCESSING PHASE

During the processing phase the following operations are performed:

(1) The compressed data generated during the preprocessing phase or from a dump tape is read into core

(2) The $CENTRAL logic is executed. Included in the $CENTRAL logic are the calls to temperature solution subroutines

(3) The specified temperature solution is performed including the following operations:

a) Prior to each temperature iteration, the $PRETEMP logic is executed

b) Following each temperature iteration but prior to pressure/flow balance on the flow systems the $POSTTEMP operations are performed

c) The pressure/flow balance is performed when required

d) The plot tape is written on the plot interval

e) The normal output is performed on the output interval. In addition, the $OUTPUT operations are performed.

Two types of subroutines are employed during the processing phase: (1) Those specifically designed for user calls in the four operation blocks and (2) those designed for internal program use. The user subroutines are described in detail in Appendix A. The subroutines designed for internal program use are summarized below.

STEP2	The main routine for the processing phase of MOTAR. It reads the compressed data tape, performs the $CENTRAL user logic and calls one or more of the following temperatures: EXPLCT, EXPSS, IMPLCT, IMPSS. The first pair or the second pair may occur in the same problem
EXPLCT EXPSS IMPLCT IMPSS	Temperature solution subroutines which are assembled during the proprocessing phase. (See Appendix A for a description of the use) These subroutines call on a number of additional subroutines which: a) Calculate the temperature network elements based upon various user options b) Perform the $PRETEMP, $POSTTEMP, and $OUTPUT user logic c) Performs temperature and pressure/flow calculations d) Writes plot and dump tape

Table I and II illustrate the possible call statements in the order that they occur for subroutines EXPLCT and IMPLCT. Only those calls needed will actually exist for a given problem. A brief description of each subroutine is given below:

TEMP2	Determines the time dependent temperatures by curve linear interpolation
PFCS	Obtains a pressure/flow balance for the input flow systems. Also calculates valve positions and pressure drop characteristics and balances the system flow with the pump flow.
INITL	Prints out initial flow rates, pressures, temperatures, capacitances, heat storage rates, thermal conductances, and time increments. Also calls PRINTS to write the initial values on the history tape.
CAPAC1	Performs linear interpolations to obtain capacitance as a function of time.
CAPAC2	Performs linear interpolations versus time and multiplies the values times constants to obtain capacitance values.

TABLE I

LISTING OF CALL STATEMENTS IN SUBROUTINE EXPLCT

```
                    } Common and Equivalence Tables

      CALL  TEMP2
      CALL  PFCS
      CALL  INITL
      ITER = 0
  100 CALL  CAPAC1
      CALL  CAPAC2
      CALL  CAPAC3
      CALL  CAPAC4
      CALL  CAPAC5
      CALL  CAPAC6
      CALL  CONDF1
      CALL  CONDV
      CALL  COND2
      CALL  COND3
      CALL  COND4
      CALL  COND5
      CALL  RADTN
      CALL  ABSHT1
      CALL  ABSHT2
      CALL  ABSHT3
      CALL  ABSHT4
      CALL  ABSHT5
      CALL  PRETMP
      CALL  KNODF
      CALL  KOND
      CALL  RADT
      CALL  DTAU1
      CALL  TEMP1
      TIME = TIME + TINC
      ITER = ITER + 1
      CALL  TEMP2
      CALL  POSTMP
      CALL  PFCS
      CALL  PRINTS
      Go to 100
                    } Writes dump tape
      RETURN
      END
```

TABLE II

LISTING OF CALLS IN SUBROUTINE IMPLCT

```
                    }
                    } Common and equivalence tables
                    }
    CALL  TEMP2
    CALL  PFCS
    CALL  INITL
100 TIMEØ  =  TIME
    TIME   =  TIME + ALPHA * TINC
    CALL  CAPAC1
    CALL  CAPAC2
    CALL  CAPAC3
    CALL  CAPAC4
    CALL  CAPAC5
    CALL  CAPAC6
    TIME  =  TIMEØ
    CALL   ABSHT 1
    CALL   ABSHT 2
    CALL   ABSHT 3
    CALL   ABSHT 4
    CALL   ABSHT 5
    CALL   IRHS
    TIME  =  TIME + TINC
    ITER  =  ITER + 1
    CALL   TEMP 2
    CALL   TEMP 1
    CALL   POSTOP
    CALL   PFCS
    CALL   PRINTS
    GØ TO 100
                    }
                    } Write dump tape
                    }
    RETURN
    END
```

CAPAC3	Obtains the capacitance of each node of this type by interpolating a time dependent curve, a temperature dependent curve and obtaining the product of the two values and then multiplying the results by an input constant
CAPAC4	Obtains the capacitance of each node of this type by interpolating a temperature dependent curve and multiplying it by an input constant
CAPAC5	Obtains the capacitance of each node of this type by interpolating two temperature dependent curves, finding the product of the two values and multiplying the results by a constant
CAPAC6	Obtains capacitance as a function one time dependent and two temperature dependent curves
CONDF1	Calculates the flow conductors for all options
CONDV	Calculates the convection conductors for all options
COND2	Obtains conductance as a function of time
COND3	Obtains the conductance by interpolating a temperature dependent curve with mean temperature between the two nodes connected and multiplying the result by a constant
COND4	Obtains the conductance between two node, one with temperature dependent properties, and one without
COND5	Obtains the conductance between two nodes, both with temperature dependent properties
RADTN	Calculates the linearized conductance due to radiation for a constant $\mathcal{F}A$
ABSHT1	Supplies constant absorbed heat values to nodes
ABSHT2	Supplies time dependent abosrbed heat values to nodes
ABSHT3	Obtains absorbed heat by interpolating a time dependent curve and multiplying by a constant
ABSHT4	Obtains absorbed heat as the product of a time dependent curve, a temperature dependent curve and a constant

ABSHT5	Obtains absorbed heat as the product of a temperature dependent curve and a constant
PRETMP	Performs user logic specified by the user in the $PRETEMP operation block
KONDF	Calculates the net heat flow to each node through flow conductors (one way) and add the quantity to the Q array Also adds the conductance values for each node to the DTAU array for future calculation of time increments
KOND	Calculates the net heat flow to each node through the normal two way conductors and adds it to the Q array. Also adds the conductance values for each node in the DTAU array for future calculation of time increments
RADT	Calculates the net heat flow to each node due to radiation conductors and adds the values to the Q array. Also adds the required constant to the DTAU array for calculating time increments
DTAU1 DTAU2	Calculates the convergent time increment for each node and applies the appropriate limits depending upon the option
TEMP1	Calculates new temperatures using the explicit method
TEMPI	Calculates new temperatures using the implicit method
POSTMP	Performs user logic specified by the user in the $ POST TEMP operation blank
PRINTS	Performs normal output, checkout print, writes history tape and calculates incrementing time values.

5.0 PROGRAM USAGE DESCRIPTION

MOTAR can be used to predict the transient or steady state temperature behavior in a system including the effects of conduction, convection, and/or radiation; to predict the pressure and flow rate for a network of tubes containing a flowing fluid for any combination of series and parallel flow; to perform any general mathematical operation by supplying the required logic to the user programming blocks including calls to the appropriate MOTAR library subroutines, or any combination of these three uses. The following is the sequence of events or steps that the user must follow in performing a thermal and/or flow analysis:

(1) Mathematical models must be constructed

(2) The values describing the elements of the models must be established and input into MOTAR in the proper format.

(3) The job must then be submitted to run on the computer system of interest following proper preparation for that system.

(4) The answers are then received for evaluation by the user.

In developing MOTAR every effort was made to permit its user to complete the above tasks with a maximum of effectiveness and a minimum of effort. The input format was designed to be as powerful as possible while remaining flexible and easy to use. The output was also designed to be flexible, giving the user the ability to choose any of the many available output options or to make his own output format when desired. Extensive error messages and check-out printing should aid the user in checking out new data decks. Extensive plotting of the output with a minimum requirement of input is available which should also add to the user effectiveness. Many other input/output options are included in MOTAR which take advantage of the available input/output devices on the NASA computer system. Included in these are the dump and restart options, the data tape and edit options, the start from the history tape options, and flux curves on tape.

5.1 MATHEMATICAL MODEL BUILDING

The first step in performing a thermal or flow-pressure analysis using MOTAR is that of building a mathematical model. The intent of this section is to describe the nature of the thermal and flow models needed for input into MOTAR.

5.1.1 Thermal Models

A mathematical model of the thermal problem must be constructed by the user so that the elements of equation (6) (Section 3.1.1) can be identified for input to MOTAR. The equation is repeated for clarity:

$$\frac{C_i \left(T_i^{n+1} - T_i^{n}\right)}{\Delta \tau} = \left[\sum_{J=1}^{nc} U_{ij} \left(T_j - T_i\right) + Q_i \right]^{m} \qquad i=1,N$$

where

C_i = the thermal capacitance of node i

= $W_i \cdot C_{pi}$

W_i = the weight of lump i

C_{pi} = the specific heat of lump i

T_i^{n+1} = temperature of lump i at iteration N + 1

T_i^{n} = temperature of lump i at iteration n

$\Delta\tau$ = the iteration time increment

U_{ij} = the overall thermal conductance between nodes i and j

Q_i = the heat rate absorbed or generated by node i

nc = the number of thermal connections

m = the time between that corresponding to iteration n and that corresponding to iteration n + 1 for evaluation of the heat flow to the node (i.e., the right side of the equation)

N = number of nodes

The primary elements that must be identified for input are:

(1) Identification numbers for each node, i

(2) Initial Temperatures for the nodes, Ti

(3) Thermal Capacitances for the nodes, Ci

(4) Absorbed or generated heat fluxes for nodes where applicable, Qi

(5) Identification numbers for the conductors and nodes connected, i and j

(6) Conductor Values Uij, whether for conducting, convection, radiation, or flowing fluid heat transfer

The thermal model of a body is constructed by subdividing it into a number of small elements of volume. By the finite difference method of solution the mass of each element is assumed to be concentrated at that nodal point which lies within the volume of the element. The temperature which is calculated represents the temperature of the point mass. Care must be taken in the subdivision of a body into its volume elements. The size of the volume elements or "lumps" must be small enough to give a sufficiently

accurate temperature distribution in the body. However, if it is excessively small a severe computer time requirement may occur. Thus, an "optimum" size is desired. A detailed discussion on the recommended practices for thermal model building will not be attempted here. Instead, the reader is refered to References 1 and 13.

An example showing the method of subdividing a body into lumps and identifying the network elements whose values must be input into MOTAR is shown in Figure 7. Figure 7(a) shows an extended fin with a base temperature, Tb, and convecting to an ambient temperature, Ta. Solar energy is incident on the fin from location l_1 to the end of the fin, l_2. Figure 7(b) shows one possible nodal breakdown for building a thermal model. It consists of 5 nodes which have been arbitrarily numbered representing equal volumes bounded by the fin temperature and the air temperature with absorbed heat on the last three nodes. Figure 7(c) shows the thermal network of the fin for the nodal breakdown of Figure 7(b). Identified on the network are the elements whose values must be input. The T's represent the node temperature, C's represent the node thermal capacitances, Q's represent the node absorbed heats, and the G's represent the conductance values between nodes which have also been arbitrarily numbered. The capacitances are calculated by

$$C(I) = (\rho)(VOL)\ (C_p)$$

where

C(I) = capacitance of node I

ρ = Density of node I

VOL = the volume of node I

Note that the boundary nodes, #6 and #7, have no capacitances since their temperatures are specified. The conductances for G(1) through G(5) are calculated by the relation for conduction which is

$$G(I) = \frac{kA}{\Delta x}$$

where

k = thermal conductivity

A = area for conduction

Δx = the conduction distance

Those for G(6) through G(10) are calculated by the relation for convection which is

$$G(I) = hA$$

where

h = convection coefficient

A = area for convection

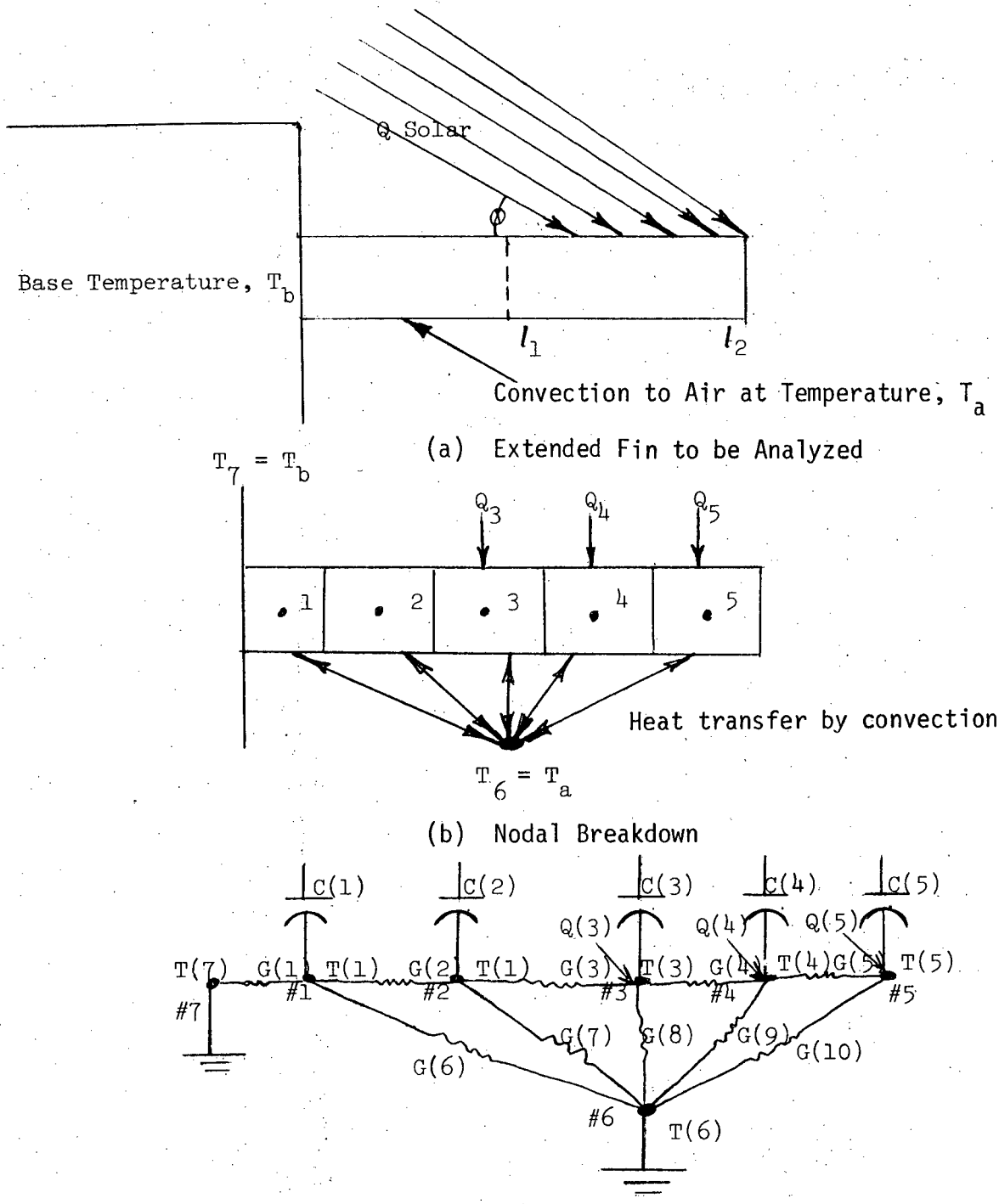


Figure 7: Example Thermal Mathematical Model

The absorbed heat for nodes 3 through 5 is calculated by

$$Q(I) = \alpha_s A Q_{solar}$$

where

Q_{solar} = Incident solar energy per unit area of the fin

A = Area of incident solar heat

α_s = Absorptivity of the fin surface to the incident heat

The input values for T's defining the initial and boundary temperature for the problem complete the identification of the elements for the problem.

5.1.2 Fluid Flow Models

A flow problem may be analyzed with MOTAR, either simultaneously with a thermal analysis, so that the flow solution is continually updated based on the thermal conditions or, the flow problem may be analyzed separately without any thermal analysis. To perform a flow analysis, the user must input a mathematical model of the flow system. The flow system is assumed to consist of a set of interconnected tubes such as the example shown in Figure 8(a) which consists of two radiator panels, each containing four tubes and connected so that they flow in parallel.

For clarity the following definitions are made at this point:

(1) A tube is any single length of pipe between two pressure nodes. A tube "contains" fluid temperature nodes and may contain as many of these as desired.

(2) A pressure node is located at each end of a tube. As many tubes as desired may be connected at a node junction and a node must exist at the junctions of two flow pipes.

We must make a mathematical model to describe the information of equation (60) to the computer. The information required consists of

(1) Identification of the pressure node numbers

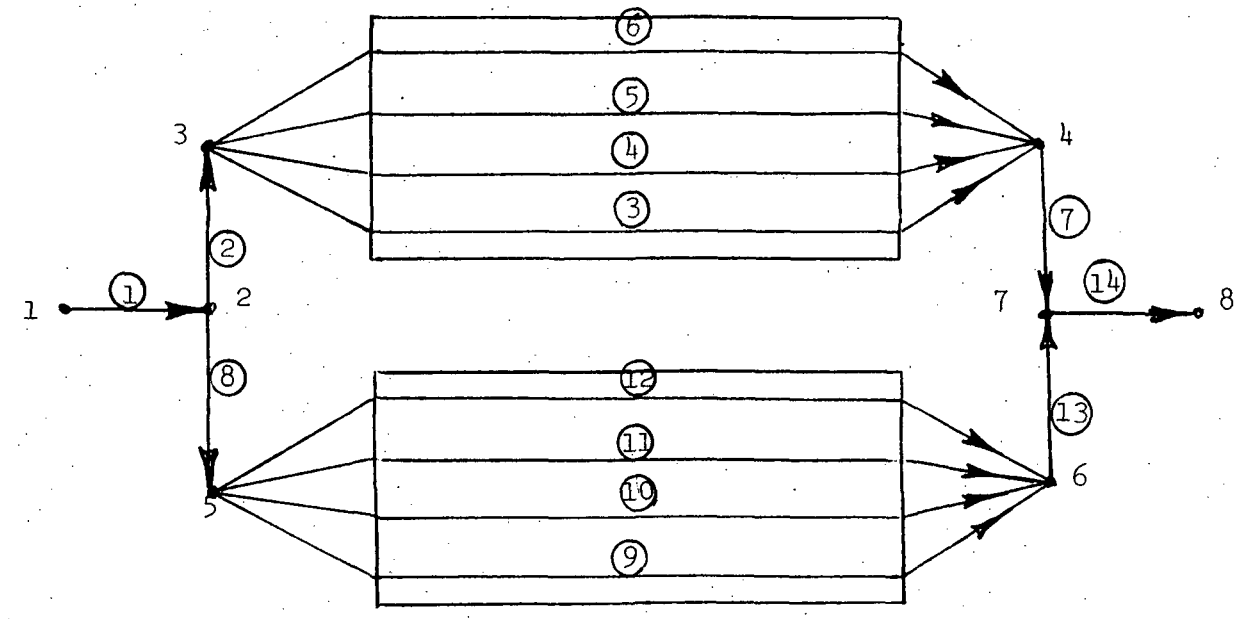
(2) Identification of the tube numbers and the two pressure nodes connected by tube

(3) The fluid temperature nodes contained in each tube

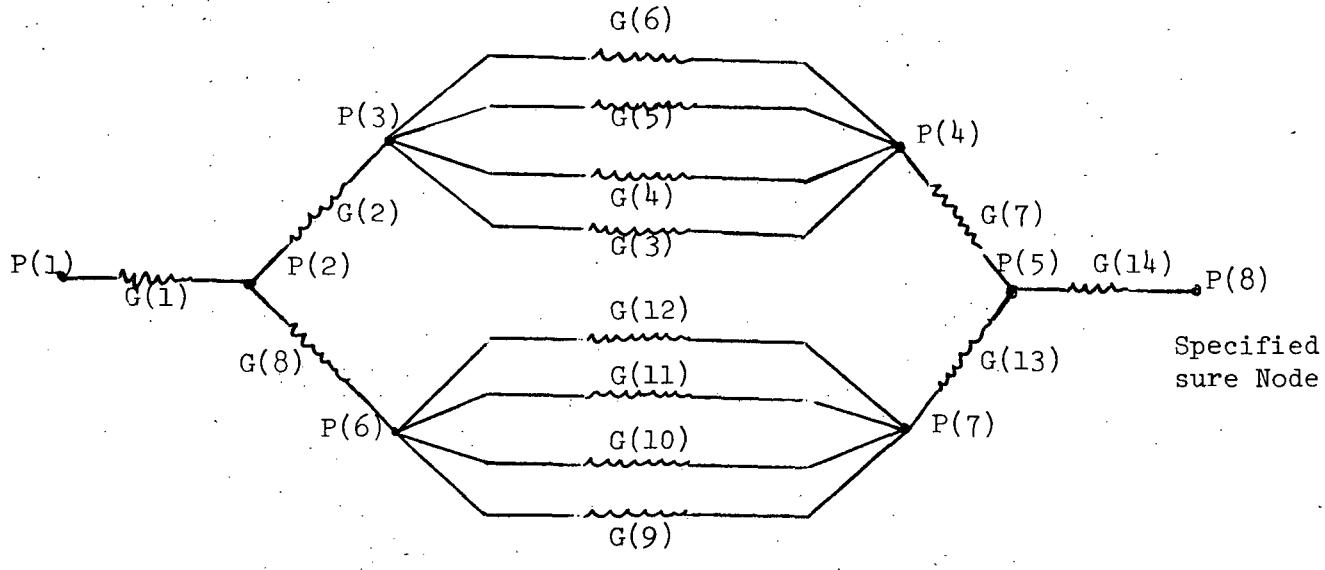
(4) The flow geometry for each temperature fluid nodes

(5) The number of "head losses" such for items such as orifices

(6) Fluid property information



(a) Flow System to be Analyzed



(b) Pressure Node/Flow Conductor Network

Figure 8 Example Flow System Mathematical Model

The fluid flow mathematical model may be described as a network equivalent to an electrical resistor network. For instance, the flow system shown in Figure 8(a) can be described by the resistance network shown in Figure 8(b). In building a flow model the user may either build a resistance/node network as shown in Figure 8(b) or simply superimpose the identifying numbers on a schematic of the flow system as shown in Figure 8(a). In either case the identifying numbers and connection information is readily available if the identifying numbers are placed on the schematic. If one adds the fluid temperature numbers to the information shown for each tube, the information for items 1, 2 and 3 above can be read directly from the schematic. Items 4, 5, and 6 must be obtained from knowledge about the system geometry and materials.

5.2 INPUT DESCRIPTION

The input for MOTAR has been designed to give the user a high degree of effectiveness and flexibility while maintaining an easy-to-use format. The effectiveness is accomplished by providing powerful options to the user which permit the input of large quantities of data with a single entry in the input. The flexibility is obtained by providing the user with a large number of options for each data entry. Many features were incorporated to make the routine easy to use which include the use of descriptive names to identify data blocks, the ability to omit blocks not requiring input data for a given problem, and the use of a free form input format. This format permits data to be input in any column between 2 and 74 (inclusive) with data values separated by delimiters.

5.2.1 General Input Requirements

The user input for MOTAR consists of three parameter cards and 7 major input data blocks with each being identified by a $ followed by its title. Some of the 7 blocks contain subordinate blocks which are identified by a + followed by the block title. Some of these subordinate blocks contain additional subordinate blocks which are identified by enclosing the block name between two apostrophies. In addition to the 7 input blocks, three additional data cards are required. The first supplies information regarding the method for data input (cards, tape, tape with data edits, etc) and information identifying whether the run is to be implicit or explicit. The second card supplies time information and the third supplies input/output option information and tolerance information for implicit runs.

A list of the possible input blocks including the major blocks and all the subordinate blocks is shown in Table III. Only those blocks required to supply information need be input. If a block is empty its block title need not be input. Also shown in Table III are three parameter cards and their location relative to the 7 Data Block. Parameter Card No. 1 is located immediately following the XQT STEP1 card and thus is the very first data card. Parameter cards no. 2 and 3 are located immediately following the XQT STEP2 Card and thus, are the very last data input cards prior to the EOF card. While all of the seven major data blocks may be input either on cards or on tape, the three parameter cards must all be input as cards and must always be supplied regardless of the type of run.

TABLE III

SUMMARY OF MOTAR INPUT DATA

```
7₈   XQT STEP 1
PARAMETER CARD 1
$    NETWORK DATA
     +  INITIAL TEMPERATURES
     +  CAPACITANCE
     +  CONDUCTORS
        'CONDUCTION'
        'RADIATION'
        'CONVECTION'
        'FLOW'
     +  ABSORBED HEAT DATA
$    FLOW SYSTEMS
        +  SYSTEM XX (CODE XX)
           'PARAMETER'
           'FLOW NETWORK'
           'SUB-NETWORK = 1'
           'SUB-NETWORK = N'
           'FLUID LUMP DATA'
           'PUMP'
           'VALVES'
        +  SYSTEM XXX (CODE XXX)
           'PARAMETER'
$    CURVES
$    CENTRAL
$    PRETEMP
$    POSTTEMP
$    OUTPUT
$    END
7₈   FOR, K STEP2
7₈   FOR, K TEMPTR
7₈   FOR, K PRETMP
7₈   FOR, K PSTTMP
7₈   FOR, K OUTPUT
7₈   FOR, K TEMPSS
7₈   XQT STEP 2
PARAMETER CARD 2
PARAMETER CARD 3
7₈   EOF
```

The general rules for MOTAR input which apply for the seven input data blocks are listed below:

1. The input for any item including data in blocks or block headings can be written in free form in any columns between 2 and 74 inclusive.

2. All blank spaces in the input are ignored. Thus, a blank card is also ignored.

3. A comment containing any character can be written on any card between an asterisk and column 74. Everything between the asterisk and column 74 will be read and written out. The comment should not extend into column 75 since this column is used for continuation. Columns 1 and 76 thru 80 are reserved for Edit. The asterisk for comments should not occur in column 1.

4. A "data group"* can be continued on the following card by any non-zero entry in column 75. This entry will cause the data group on the card prior to the asterisk (if one exists) to be continued on the next card.

5. Several data groups may be entered on a single card if separated by a slash (/). For example three initial temperatures (each being a single data group) might be put on one card as follows:

   1 = 90./5 = 100. /7 = 97.5

6. Any system of units can be used for a given problem but all input items must be in consistent units. This required the following values be input in the problem units: (1) the temperature for the input units at absolute zero and the Stephan-Boltzman constant for thermal radiation analyses and (2) the gravitational constant (g_c) for flow-pressure analysis.

7. The seven data blocks may be input in any order. All subordinate blocks may be input in any order as long as they are input under their main heading.

---

* A "data group" is defined as a group of data which must be input in a specified order, For instance, the input required for a conductor would be a data group consisting of the conductor number, the two nodes connected and the value of the conductance.

8. The heading or title for each data block or subordinate block is entered with the proper delimeter ($, +, or ') followed by the block title or at least the first three letters of the title (May be input in any column between 2 and 74 inclusive)

9. Any block or subordinate block not required can be omitted including its heading card if it supplies no information.

10. Numbers may be input as integers or real numbers. The real numbers may be input in pure decimal form such as 3.54 or in exponential form such as .376E-10 where the exponent is -10. As with all other input, blanks in numbers are ignored so that 3. 54 would be read 3.54. Also, the user may input as many significant figures as desired and the computer will pick up to the maximum digit capability of the machine.

The input requirements for the parameter cards, the 7 major data blocks, and their subordinate blocks, are described in the following subsections.

5.2.2 Parametric Data Card

Three parametric data cards must be supplied on each MOTAR run to supply data edit information, time information, input/output codes and implicit run information. Unlike the seven data blocks, these cards must always be supplied and the data must be entered in a fixed format. The location of the parameter cards relative to the other input data and system setup cards is shown in Table III. A description of the input for each of the three parameter cards is given below:

PARAMETER CARD NO. 1

COLUMNS	FORTRAN NOMENCLATURE	FORMAT	DESCRIPTION
1-5	INDATA	I5	=0, All data supplied on cards =1, Card images are written on Unit B =2, Card edits with Unit C used to generate unit B =3, Use unit B without edits =-2, Same as 2 except data from Unit B will be punched =-3, Same as 3 except data from unit B punched
6-10	MPLCT	I5	=0, Explicit method of solution ≠0, Implicit method of solution

PARAMETER CARD NO 2

1-10	TIME	F10.0	Problem Start Time

COLUMNS	FORTRAN NOMENCLATURES	FORMAT	DESCRIPTION
11-20	TINCMN	F10.0	Minimum stable time increment for the No Overriding Option. TINCMN must be 0 for the Override or Floating Option.*
21-30	TINC	F10.0	Problem Time Increment
31-40	STIME	F10.0	Problem Stop Time
41-50	WINC	F10.0	Print Interval
51-60	PINC	F10.0	Plot Interval. If 0, no plotting will occur
61-70	TMPTIM	F10.0	Time that history tape will be read for initial temperatures, flow rates and value position. If 0, will be set to TIME.
71-80	DQTIME	F10.0	Time scale shift for flux curves. DQTIME is added to the problem time prior to interpolating flux cruves.
PARAMETER CARD NO. 3			
1-5	RTIME	F 5.0	Computer Time in minutes requested for run if 0., will be set to 5.
6-7	ISTART	I2	= 0, This problem is not being restarted Data must be supplied for starting problem ≠ 0, This problem is being restarted Restart data is supplied on Unit L
8-9	NEWTMP	I2	= 0, The history tape will not be read to obtain initial conditions ≠ 0, The history tape, Unit H, will be read at TMPTIM (Card 2) to obtain initial conditions
10-11	NFLXCD	I2	= 0, No incident heat curves will be supplied on separate flux tape ≠ 0, Some of the incident heat curves will be supplied on Unit E
12-13	NCKOUT	I2	= 0, No checkout print will be given ≠ 0, A checkout print will be given
14-15		Blank	
16-20	MXPASS	I5	Maximum number of Gauss-Siedel iterations permitted on each iteration for solution to temperature equations. If = 0, set to 100

* See Page A-5

COLUMNS	FORTRAN NOMENCLATURES	FORMAT	DESCRIPTION
21-25	ALPHA	F5.5	Point within the iteration for evaluation of heat flux on implicit runs. If ALPHA = 1.0, backward difference occurs; if ALPHA = 0.5, mid difference occurs; if ALPHA = 0 it is set to 1.0; if ALPHA < 0.5 it is set to 0.5.
26-30	DTMXA	F5.5	Temperature Solution Tolerance. A solution is reached when all temperatures change less than DTMXA on a given iteration. When =0, set to 0.01
31-35	ORP	F5.4	Overrelaxation parameter for implicit solution. Set to 1.0 if 0.
36-45	SSTEST	F10.9	Steady state test. Steady state is reached when all temperatures change less than SSTEST on a given iteration; if SSTEST = 0.0, it is set to 0.0001
46-55	IPASS	I10	Maximum number of iterations permitted to reach steady state; if IPASS = 0, it is set to 10000.

5.2.3 Network Data Block

The "NETWORK DATA" block contains the information required to describe the thermal network. The information is input in four subordinate blocks with the conductor block containing four additional subordinate blocks as shown in Table III. The following subordinate blocks are continued in the NETWORK DATA block:

```
$    NETWORK DATA
     +    INITIAL TEMPERATURES
     +    CAPACITANCES
     +    CONDUCTORS

          'CONDUCTION'
          'FLOW'
          'RADIATION'
          'CONVECTION'

     +    ABSORBED HEATS
```

The blocks headed by + may be input in any order and those enclosed in apostrophes may be in any order but must all be contained in the conductor block. Any block or subordiante block not required may be omitted. A description of the input requirements and options for each of the blocks is described below.

5.2.3.1 Initial Temperatures

The initial temperatures are supplied in the block headed by:

```
     +    INITIAL TEMPERATURE
```

The format for input of the initial temperatures is an integer, or a group of integers, representing node numbers followed by an equal sign followed by the specification of the initial temperature. Thus, the numbers on the left side of the equal specify node numbers and numbers on the right side specify temperatures. Several options are available for both sides.

The option for specifying the node numbers on the left of the equal sign are described below.

OPTION 1: Single input

The input format is:

NN = TI

where NN = The node number

TI = the node initial temperature

If NN is a negative integer the node is a boundary and no calculation will be made to change the temperature. Thus, it will remain at the initial value, TI throughout the problem. If an integer is supplied instead of a real number, it specifies a time dependent curve which describes the temperature of node NN as

a function of time.

OPTION 2: Multiple input separated by commas

The input format is:

N1, N2, ---, NN = TI

where N1, N2, ---NN = the node numbers with initial temperature, TI

As many node numbers as desired by be supplied in this manner (see Section 5.2.1 for rules on continuation cards, etc.). The node numbers, N_1, N_2 ---NN, may be input in any random order.

OPTION 3: Input of a group of sequential nodes

The input format is:

N1 thru N2 = TI

where N1 is the starting node number of the group

N2 is the final node number of the group

This option means all nodes between and including N1 and N2 are assigned a value of TI. N1 may be either smaller than, larger than, or equal to N2. It may be an integer or real as described on Option No. 1.

OPTION 4: Input of a group of nodes with equal spacing between the numbers

The input format is:

N1 thru N2 by N3 = TI

where N1 = the starting node number of the group

N2 = the final node number of the group

N3 = the integer spacing for the numbers between the starting and final numbers

Using Option 4, N2-N1 must be an integer multiple of N3.

Some illustrative examples of the input for the initial temperatures are shown in Table IV.

5.2.3.2 Thermal Capacitances

The thermal capacitances for the temperature nodes are input in the block headed by:

TABLE IV

EXAMPLES OF INPUT FOR INITIAL TEMPERATURES

+ INITIAL TEMPERATURES	
1 = 50	* Node 1 is initially at temperature = 50.
-3 = -459.69	* Node 3 is a boundary at -459.69
2, 4, 5 = 75.	* Nodes 2, 4, and 5 initially at 75.
6 = 11	* Node 6 temp supplied on curve 11.
7 = 62. / 8= 43. /9 = 77	* Initial temperature of nodes 7, 8, and 9 all supplied on the same card.
10 thru 19 = 102.	* Nodes 10 thru 19 all have initial temps of 102.
51 thru 55 = 21	* Temperature vs. time for nodes 51 thru 55 supplied on curve 21.
40, 42 thru 45, 74 = 100.	* Initial temperatures of node 40, 42 thru 45, and 47 are 100.
22 thru 38 by 2 = 57.	* Temps for even nodes between 22 and 38.
20, 21 thru 39 by 2, 56 thru 60 = 66.	* Temps for node 20, odd nodes between 21 and 39 and 56 thru 60 = 66.

+ CAPACITANCE

which is subordinate to the NETWORK DATA block. The format for input of the thermal capacitance is specification of the node numbers on the left of an equal and specification of the capacitance on the right of the equal. The options for specifying the node numbers are the same as those for the initial temperatures described in section 5.2.3.1. The values for capacitance are specified to the right of the equal in either one, two or three values. Any of the values (whether one, two or three values are supplied) may be integers or real numbers (containing decimals). When integer values are supplied, each identifies a tabulated curve to be interpolated during the run. An integer for the first value to the right of the equal identifies a curve which is a function of time. Integers for the second and/or third values indicate curves to be interpolated as a function of the node temperature. Any combination of one, two, or three numbers,any of which may be real or integers, may be input. In the preprocessing phase (problem setup) all real numbers are multiplied together. In the processing phase (the problem analysis) real values are obtained from the curves identified by integers and multiplied times the constant values to obtain capacitances on each iteration.

Some examples for capacitance input are shown in Table V..

5.2.3.3 Thermal Conductors

Thermal conductors are input in a block headed by

+ CONDUCTORS

which is contained in the higher level $ NETWORK DATA block. The + CONDUCTORS blocks data is contained in four subordinate blocks headed by 'CONDUCTION', 'CONVECTION', 'FLOW', and 'RADIATION', each containing the input for the type of conductor indicated by its descriptive heading. Some of the general rules which apply to conductors are:

(1) A conductor number must be unique regardless of the type of conductor. That is, a conductor number used in one block cannot again be used in that block or any of the other three blocks.

(2) Conductor numbers do not have to be input in any particular order. Also, they do not have to be numbered sequentially although sequential numbering is most efficient from a space utilization standpoint.

(3) The four conductor subordiante blocks may be input in any order as long as they are within the +'CONDUCTOR'heading.

(4) The headings for each of the conductor subordinate block consists of their name enclosed in quotations as follows:

(a) 'CONDUCTION'
(b) 'CONVECTION'
(c) 'FLOW'
(d) 'RADIATION'

The nine general rules previously given for input also apply.

The input format for the conductors consists of connection identification numbers (conductor numbers, and nodes connected) on the left side of the equal and specifications for determining the conductor value on the right of the equal. The options for specifying connections identification

TABLE V
EXAMPLES OF CAPACITANCE INPUT

```
+ CAPACITANCES
1                          =  .35           *  C(1)=.35
11,12,13                   =  .47,.06,.53   *  C(11)=C(12)=C(13)=.47X.06X.53
3 THRU 10                  =  .61,.27       *  C(3)---C(10)=.61X.27
12                         =  .32,62.4,22   *  C(12)=.32X62.4XCURVE(22,T(12))
13 THRU 17 BY 2            =  .32,12,22     *  C(13)=C(15)=C(17)=.32XCURVE
                                            *  (12,T)XCURVE(23,T)
14 THRU 18 BY 2            =  .07,12,.5     *  C(14)=C(16)=C(18)=(.07)(CURVE
                                            *  (12,T)X.5
19 THRU 25,31 THRU 35      =  .57,12        *  C(14)---C(25),C(31)---C(35)
                                            *  =.57XCURVE(12,T)
26 THRU 30                 =  41,.06,70.    *  C(26)---C(30)=CURVE(41,TIME)
                                            *  X.06*70.
36                         =  41,5.3        *  C(36)=CURVE(41,TIME)X5.3
37                         =  51            *  C(37)=CURVE(51,TIME)
38                         =  61,12,22      *  C(38)=CURVE(61,TIME)XCURVE
                                            *  (12,T(38))XCURVE(22,T(38))
39                         =  61,12,.76     *  C(39)=CURVE(61,TIME)XCURVE
                                            *  (12,T(39))X.76
40                         =  61,70.,22     *  C(40)=CURVE(61,TIME)X70.X
                                            *  CURVE(22,T(40))
41                         =  61,12         *  C(41)=CURVE(61,TIME)XCURVE
                                            *  (12,T(41))
```

In the comments above,

C(N)	=	Capacitance of node N
CURVE(M,T(N))	=	the interpolated value of curve M at the temperature of node N
CURVE(M,TIME)	=	the interpolated value of curve M at the problem time
X	=	indicates multiplication

are the same for all four conductor blocks and are similar to those for specifying node numbers in the + INITIAL TEMPERATURE block except the numbers are input in groups of 3. These three numbers are conductor number, node connected, node connected. The options for specifying the conductors values are numerous and vary depending on which of the four subordinates blocks the input is in. The various options for connections identification and connection value specification are described below:

Conductor Connections Identifications

The connections identification for the conductors are input on the left of an input equal sign and they identify the conductor number and nodes connected. The options are the same for these identifiers (left side of the equal) for all blocks subordinate to the +CONDUCTOR heading. The options for connection identifications are described below:

OPTION 1: Single connection input

The input format is:

NC, N1, N2 = VALUE

Where NC = the conductor number
N1,N2 = Nodes connected
VALUE = The specified value of the conductor (many options are avaiable depending on the type of conductor

The order of input of N1 and N2 is important for the'FLOW' block and 'CONVECTION' block.

OPTION 2: Multiple input separated by commas

The input format is:

NC1,N11,N21,NC2,N12,N22,---NCn,N1n,N2n = VALUE

Where NC1,NC2 ----NCn = Conductor numbers
N11, N21 = First set of nodes connected
.
.
.
N1n,N2n = nth set of nodes connected

Any option for determining VALUE may be used which is avaiable in the input block under consideration.

OPTION 3: Input of a group of sequentially incremented connections

The input format is:

NC1,N11,N21 THRU NCn,N1n,N2n = VALUE

Where NC1 = the starting conductor number
N11 = the starting node number for side 1
N21 = the starting node number for side 2
NCn = the final conductor number
N1n = the final node number for side 1
N2n = the final node number for side 2

For this option, NCn-NC1 must equal N1n-N11 and N2n-N21. All the options available for VALUE are applicable here.

OPTION 4: Input of a group of connections sequentially incremented by an input integer

The input format is:

NC1,N11,N21 THRU NCn,N1n,N2n BY IC,I1,I2 = VALUE

Where NC1 = the starting conductor number
N11 = the starting node number on side 1
N21 = the starting node number on side 2
NCn = the final conductor number
N1n = the final node number on side 1
N2n = the final node number on side 2
IC = the increment for the conductor numbers
I1 = the increment for the node numbers or side 1
I2 = the increment for the node numbers on side 2

For this option, the following relations must hold:

$$\frac{NCn - NC1}{IC} = \frac{N1n - N11}{I1} = \frac{N2n - N21}{I2}$$

When $I_X = 0$, N_Xn must equal N_X1 for the nodes connected

'CONDUCTION' BLOCK

Numer options are available specifying the conduction conductor values or methods for determing their values. These specifications are input on the right of the equal sign with any of the connections identification options discussed above on its left. The options for conduction input are described below.

OPTION 1: Constant Conductor

The input format is:

NC, N1, N2 = CONST

Where NC,N1,N2 = Connections identification input by any of the available options
CONST = the constant value of the conductance

OPTION 2: Conduction in homogeneous material which is tem-

perature dependent. Input by a temperature dependent curve with a constant multiplier The input format is:

NC,N1,N2 = AX, NK (TM)

Where NC,N1,N2 = connections identifications input by any available option

AX = the constant portion of the conductance typically A/X, where A = conduction area and X = distance

NK (TM) = a temperature dependent curve number which is interpalated with temperature TM. Normally this is a thermal conductivity curve

TM = Mean temperature between T(N1) and T(N2)

$$= \frac{T(N1) + T(N2)}{2.0}$$

OPTION 3: Conduction in non-homogeneous material where one material is temperature dependent and one is not.

(A) N1 is the node containing temperature dependent thermal conductivity.

The input format is:

NC,N1,N2 = AX1, NK(TM), KAX2

Where NC,N1,N2 = connections identification input by any of the available options

AX1 = A/X for node N1

NK(TM) = thermal conductivity curve for the materials of node N1 interpolated on each iteration at the temperature of N1

KAX2 = the constant value of KA/X for node N2

The conductance for each iteration is obtained by

$$U(NC) = \frac{1}{\frac{1}{AX1\ K(TN1)} + \frac{1}{KAX2}}$$

(B) N2 is the node containing temperature dependent thermal conductivity

The input format is:

NC,N1,N2 = KAX1,AX2,NK(TN2)

Where NC,N1,N2 = Connections identification input by any of the available options

KAX1 = The constant value for conductance, KA/X for node 1

AX2 = A/X for node N2

NK(TN2)= the thermal conductivity curve number for the material of node N2 interpolated at temperature of node N2

OPTION 4: Conduction is a non-homogeneous material with that of both nodes being temperature dependent

The input format is:

NC,N1,N2 = AX1,NK(TN1),AX2,NK(TN2)

Where NC,N1,N2 = Connections identifications input by any of the available options

AX1 = A/X for node N1

NK(TN1) = the thermal conductivity curve number of node 1. Material interpolated at temperature TN1

AX2 = A/X for node N2

NK(TN2) = the thermal conductivity curve number of node 2. Material which is interpolated at temperature

The conductance is determined on each iteration when using this option by

$$U(NC) = \frac{1}{\frac{1}{(AX1)k(TN1)} + \frac{1}{(AX2)k(TN2)}}$$

OPTION 5: Conductance is a function of time
The input format is:

NC,N1,N2 = NKAX (TIME)

Where NC,N1,N2 = connections identification input by any of the available options

NKAX(TIME) = An iteger specifiying the value of the conductance as a function of TIME

Some examples of the 5 options for specifying the conductance values for conduction are given in Table VI.

TABLE VI
EXAMPLES OF CONDUCTION INPUT

```
'CONDUCTION'

1,5,6=1.3                    *  CONDUCTOR NO.1 HAS A CONSTANT VALUE OF 1.3

2,6,7=.8,62                  *  A/X=0.8;K INTERPOLATED AT (T(6)+T(7))/2. ON
                             *  EACH ITERATION; CONDUCTANCE = KA/X

5,7,8=.19,62,11.             *  A/X(7)=.19,K(7)=CURVE(62,T(7))),
                             *  KA/X (8)=11.

7,20,21=11.,.19,62           *  KA/X(20)=11.,A/X(21)=.19,K(21)=CURVE
                             *  (62,T(21))

6,,7,20=.19,62,.31,72        *  A/X(7)=.19,K(7)=CURVE(62,T(7))A/X(20)=.31,
                             *  K(20)=CURVE(72,T(20))

9,21,8=11                    *  CONDUCTANCE NO 9 GIVEN AS FUNCTION OF TIME
                             *  ON CURVE 11
```

In the above

A	=	Area for conduction
K	=	Thermal conductivity
X	=	Conduction Distance
CURVE(M,T(N))	=	Interpolation of tabulated curve no. M at the temperature of node N,T(N)

'CONVECTION' Block

The options which are available for determining conductor values for convection heat transfer are primarily options for determining the convection heat transfer coefficient, h, since the conductor values is hA, A being the constant heat transfer area for convection. A is input as a constant value for all the convection options. As described in Section 3.1.3.1 options available for calculating h are (repeated here for clarity)

Option 1: h for flow in a tube

(A) LAMINAR FLOW ($Re \leq 2000$)

$$h = \frac{k}{D} \quad 3.66 \cdot F1 + \frac{.0155 \cdot F2}{\frac{1}{RePr} \cdot \frac{X}{D} + .015 \left(\frac{1}{RePr} \cdot \frac{X}{D}\right)^{1/3}}$$

(B) TRANSITION FLOW ($2000 < Re < 6400$)

$$h = \frac{k}{D} \quad [.116 \, (Re^{2/3} - 125) \, (Pr)^{1/3}]$$

(C) TURBULENT FLOW ($Re \geq 6400$)

$$h = .023 \frac{k}{D} \; Re^{.8} \; (Pr)^{1/3}$$

Option 2: $St \, Pr^{2/3} = f(Re)$ or $Nu = f(Re) \, Re \, Pr^{1/3}$

$$h = \frac{k}{D} \; [f(Re) \, Re \, Pr^{1/3}]$$

Option 3: Heat transfer coefficient is a function of flow rate

$$h = f(\dot{w})$$

Option 4: Heat Transfer Coefficient is a function of time

$$h = f(t)$$

The nomenclature for the above equations is as follows:

h = the convection heat transfer coefficient

k = the thermal conductivity of the fluid

D = the hydraulic diameter of the flow passage

x = the flow length from the start of the tube

Re = Reynolds number

Pr = Prandtl's number

F1 = Laminar Flow fully developed factor (input)

F2 = Laminar Flow entry length factor (input)

f(Re) = A tabulated curve of $ST(Pr)^{2/3}$ vs Re or $NU/(Re\ Pr^{1/3})$ vs Re

f(t) = A tabulated curve of h vs time

The MOTAR input for the convection block is headed by the input of 'CONVECTION'. The information needed for the various options is described below.

Input for Option 1:

NC, NF, NT = AHT, NTUBE, F1,F2

where

NC = Conductor number

NF = fluid lump number

NT = tube lump number

AHT = Area for heat transfer

NTUBE = tube number for finding flow rate and fluid properties

F1 = Laminar fully developed factor

F2 = Laminar entry length factor

The values for F1 and F2 are assumed to be 1.0 if they are omitted.

Input for Option 2:

NC, NF, NT = AHT, TUBE, CURVE ($Re, ST\ Pr^{2/3}$)

Where symbols are the same as above except

CURVE($Re, STPr^{2/3}$) = a type 4 curve number which gives $STPR^{2/3}$ as a function of Re (See Section 5.2.4 for curve data description).

St = Stanton number

Re = Reynolds number

Input for Option 3:

NC, NF, NT = AHT, TUBE, CURVE ($\dot{w}$,h)

Where symbols are the same as for Option 1 except

CURVE ($\dot{w}$,h) = a type 3 curve number of a curve which gives h as a function of $\dot{w}$ in the tube

h = heat transfer coefficient

$\dot{w}$ = flow rate of fluid in the tube

Input for Option 4:

NC, NF, NT = AHT, CURVE (t,h)

Where symbols are the same as those for Option 1 except

CURVE (t,h) = A type 1 curve number of a curve which gives h as a function of t

h = heat transfer coefficient

t = time

All of the options previously described for group input of the connections identification (NC, NF, NT) are applicable to convection conductor input.

'FLOW' Block

The inputs to define the flow conductor connections and methods for their calculations are supplied in the conductor subordinate block headed by 'FLOW'. Flow conductors are "one-way" conductors designed to simulate the flow of fluid in a tube. The conductors can be utilized in other applications however. The flow conductor is calculated by

$$U_{ij} = \dot{w}_i \, C_{Pi}$$

Where U_{ij} = the conductance from node i to node j (but not back the other direction)

$\dot{w}_i$ = the fluid flow rate

C_{Pi} = the fluid specific heat

The input for flow conductors is the same as for the other conductors on the left of the equal sign. That is, the same options apply for the connections identification input. The input which specifies the options and values to be used in calculating conductor values is supplied on the right of the equal. It consists of four input values: KODEF, FVALUE, KODEC, CVALUE as follows:

NC,NFL,NTL = KODEF, FVALUE, KODEC, CVALUE

Where
- NC = conductor number
- NFL = fluid lump number
- NTL = tube lump number
- KODEF = fluid flow code
- FVALUE = value or location of the value of flow rate
- KODEC = specific heat code
- CVALUE = value or location of the value for specific heat

The method for determining the flow rate is specified by KODEF and FVALUE. The flow options are described in the tabulation below:

KODEF	FVALUE
1	Constant value specifying the flow rate
2	Curve number of a curve which supplies flow as a function of time
3	Tube number in flow network data from which flow can be obtained

The method for determining the specific heat is specified by the values of KODEC and CVALUE. The various options are described in the tabulation below:

KODEC	CVALUE
1	Constant value for the specific heat
2	Curve number of a curve which gives specific heat as a function of temperature of NFL
3	Flow system number from which the fluid data can be found

Any combination of KODEF/FVALUE can be used with any combination of the KODEC/CVALUE.

'RADIATION' block

The radiation conductor connections and values are specified in the conductor subordinate block headed by

```
"RADIATION'
TZERO=-XXX.XX          (XX = numbers)
SIGMA=XX.XX
```

Where TZERO = the temperature at absolute zero for the problem temperature scale (always negative or zero)

SIGMA = the Stefan-Boltzman constant in the problem units

The conductor inputs consists of the connections identifications on the left of the equal sign and specifications to identify values of FA on the right of the equal. The options for the connections are the same as all the other conductor options and have been previously discussed in this section. The options for specifying the values for FA are described below

Option 1: Constant FA

Input format:

NC, N1, N2 = FA_{12}

Where NC = Conductor number

N1, N2 = Nodes connected

FA_{12} = FA value between nodes 1 and 2 (constant)

Option 2: Constant A with temperature dependent supplied on a curve

Input format:

NC, N1, N2 = A_1, NF_{12} (T(N1))

Where A_1 = radiator area for N1

NF_{12}(T(N1)) = temperature dependent curve which supplies F from N1 to N2 as a function of T(N1)

Option 3: Curve of FA as a function of time

Input format:

NC, N1, N2 = NFA_{12}(TIME)

Where $NFA_{12}(TIME)$ = a curve number which supplies FA as a function of time

The FA_{12} is determined by one of the three options above. The conductance is then determined by

$$U_{12} = (SIGMA)(FA_{12})[(T_1 - TZERO)^2+(T_2 - TZERO)^2)]$$
$$[(T_1 - TZERO) + (T_2 - TZERO)]$$

5.2.3.4 Absorbed Heats

Absorbed and/or internally generated heat for each node is input in the block headed by:

+ ABSORBED HEAT

This block is subordinate to the $ NETWORK DATA block. The general format for the input consists of (1) the identification of the node numbers on the left of an input equal sign and (2) specification of the option to be used to obtain the absorbed heat for the node on the right of an equal. The options available for identification of node numbers on the left of the equal are the same as those for initial temperatures as discussed in section 5.2.3.1.

The methods for calculating absorbed heat are specified to the right of the equal sign by either one, two or three numbers separated by commas. The first number may be either real or integer. An integer specifies a time dependent curve number which is interpolated on each iteration. A real number is simply a multiplier for other values if others exist. The second number may also be input as a real number or an integer. An integer for the second number specifies a temperature dependent curve number which is interpolated at the node temperature on each iteration. A real number for the second number is a constant multiplier. The third number, if it exists, must be a real constant and is a multiplier.

The format for specifying the absorbed heat using the various available options are described below.

Option No. 1: Absorbed heat is one constant, two constants or three constants

The input format is

NN = QI, ALP,A

where NN = the identification of node numbers by any available option discussed in section 5.2.3.1.

QI,ALP,A = input real constants which are multiplied together to obtain absorbed heat. The second and third constants are optional; i.e., there may be one, two, or three constants

Option No. 2: The absorbed heat is a function of time with or without constant multipliers

The input format is

NN = NQI(TIME), ALP,A

Where

NN	=	The identification of node numbers by any available option
NQI(TIME)	=	an integer specifying a curve number of a time dependent variable
ALP, A	=	Input real constants, and are both optional; i.e., either zero, one, or two real constants may be input

The absorbed heat is determined on each iteration by interpolating the NQI curve and multiplying the value by ALP and A.

Option No. 3: The absorbed heat is a function of temperature with or without constant multipliers.

The input format is

NN = QI, NALP (T(NN)), A

Where

NN	=	The identification of node numbers by any available option
NALP(T(NN))	=	An integer specifying a curve number of a temperature dependent curve
QI,A	=	Input real constants, A is optional but, QI must be supplied (May be 1.0 if not needed)

The absorbed heat for nodes NN are determined by interpolating curve NALP at temperature of node NN on each iteration, multiplying the value by QI and by A if available.

Option No. 4: The absorbed heat is a function of both temperature and time with or without a constant multiplier.

The input format is

NN = NQI(TIME), NALP(T(NN)),A

Where	NN	=	The identification of node numbers by any of the available options
	NQI(TIME)	=	An integer specifying a curve number of a curve of a time dependent variable
	NALP(T(NN))	=	An integer specifying a curve number of a curve of a temperature dependent variable
	A	=	An optional real multiplier

The absorbed heat is determined by interpolating curve NQI at time and NALP at the temperature of node NN on each iteration, multiplying the interpolated values and multiplying this product by A if it is available.

The absorbed heat calculated by any of the above options is added to that already calculated by some other means. Thus, a user may specify as many absorbed heat curves as required.

5.2.4 Flow System Data

If the problem being analyzed by MOTAR requires a pressure/flow analysis, the data needed to describe the flow system is input in the block headed by the title,

```
$  FLOW SYSTEMS
```

The input values in this block may be in any system of units, but the units must be consistent throughout both the thermal and fluid portions of the problem. In order to permit this, the user must supply the gravitational constant immediately following the heading as follows:

```
     GC = 32.174
or   GC = 3.2174E1        (For feet and seconds)
```

Table VII gives values of GC for various units of length and time.

The $FLOW SYSTEMS block data is contained in the following subordinate blocks:

```
$ FLOW SYSTEMS

+  SYSTEM = 1 (KODE 1)
   'PARAMETER'
   'FLOW NETWORK'
   'SUBNETWORK = 1'
      '     '     '
      '     '     '
      '     '     '
   'SUBNETWORK = NSN'
   'FLUID LUMP DATA'
   'PUMP'
   'VALVES'
+  SYSTEM = 2 (KODE 2)
   'PARAMETER'
   'FLOW NETWORK'
   'SUBNETWORK = 1'
      '     '     '
      '     '     '
      '     '     '
   'SUBNETWORK = NSN'
   'FLUID LUMP DATA
   'PUMP'
   'VALVES'

+  SYSTEM = 3
    '
    '
   etc.
```

TABLE VII VALUE OF GC FOR VARIOUS PROBLEM UNITS

UNITS				GC
MASS	FORCE	LENGTH	TIME	
LB	LBm	In.	Sec	386.1
			Min	1.390×10^{6}
			Hr	5.004×10^{9}
		Ft.	Sec	32.174
			Min	1.1583×10^{5}
			Hr	4.1696×10^{8}
		Yd.	Sec	10.725
			Min	3.861×10^{4}
			Hr	1.3899×10^{8}
GRAM	dyne	Centimeter	Sec	1.0
			Min	3600.
			Hr	1.296×10^{7}
KILOGRAM	Newton	Centimeter	Sec	1×10^{-2}
			Min	36
			Hr	1.296×10^{5}
		Meter	Sec	1.0
			Min	3600.
			Hr	1.296×10^{7}

The $ FLOW SYSTEM block may be omitted if no flow analvsis is required.

A flow system is defined here as a single set of interconnected tubes which contain a fluid which can be described by a single set of properties. Any number of flow systems may be analyzed in the same problem but each required a separate set of data headed by the card

+ SYSTEM = SN (KODE)

Where SN = an integer describing the system number

KODE = a code indicating whether the system is one phase flow or two phase flow

The system number must be unique for each system but consective numbering isn't reuqired. For the current version of MOTAR, the two phase flow capability is not included and thus, KODE is always a value of 1. Four subordinate blocks are always required for each system. These blocks, which are 'PARAMETERS', 'FLOW NETWORK', 'FLUID LUMP DATA', and 'PUMP' supply information which is always required to define a system. The 'VALVES' and the 'SUBNETWORK' blocks are included only when they supply additional information. The input required for each of the flow systems subordinate blocks is described below.

5.2.4.1 Parameters

The parameters block must be supplied for each flow system. This block is headed by the card

'PARAMETERS'

and contains the following information:

1. Specification of the following fluid properties:
   (a) thermal conductivety
   (b) density
   (c) viscosity
   (d) specific heat
2. The node number and pressure value of the reference pressure node.
3. Parameter values describing the characteristics for the pressure-flow solution.
4. When a numerical solution rather than the direct solution for the pressure equations desired, a solution tolerence is required. This also a key input indicating a numerical solution is desired.

The above items are input by supplying a variable name on the left of an equal and the specification of its value on the right. The following is a description of the variable names

Fluid Properties

KT : Fluid Thermal Conductivity

RO : Fluid Density
MU : Fluid Viscosity
CP : Specific Heat

The value for each of the fluid properties may be specified as a constant by supplying a real value or it may be specified as a temperature dependant curve by supplying the value as an integer. The integer value must be the curve number. For example, if all the values were constant they would be specified as,

KT = .25 / RO = 67. / MU = .137 / CP = .31

If instead, the thermal conductivity and viscosity were supplied by a temperature dependent curve, the values would be supplied by,

KT = 32 / RO = 67. / MU = 42 / CP = .31

These values must always be supplied for each system.

Reference Pressure Specification

The pressure at the pump inlet node for a closed system or for the system outlet for an open system must be specified. This is supplied in the following format:

P(NN) = VALUE

Where NN = an integer representing the pressure node number
VALUE = a real value representing the pressure value of the node

For example:

P(10) = 14.7

indicates pressure node number 10 is the reference pressure node number with a value of 14.7.

Pressure-Flow Solution Parameters

Four optional variables may be supplied by the user in the 'PROPERTIES' block to define certain characteristics of the solution. These are described below:

MPASS: An integer specifying the number of temperature iterations between balancing of the pressures and flow rate. It is set to one if not supplied.

MXPASS: The maximum number of tries to balance the pressures and flow rates for each iteration. Routine sets to 100 if it is not supplied.

TOL: The tolerence for terminating the pressure/flow balance. When the fraction of change of the flow rates in all the tubes is within TOL, the solution is reached. If TOL is not supplied, it is set to 0.001.

A: An averaging factor to help improve the convergence rate for flow systems that are very nonlinear - ie, system with turbulent flow or head losses. It must be a value greater than 0.0 and less than
The averaging factor is applied to each tube flow rate following each pass through the pressure/flow balancing loop as follows:

$$\dot{W}_{new} = A\,\dot{W}_{new} + (1.-A)\,\dot{W}_{old}$$

If A is not supplied, it is set to 0.7.

FLOW: The initial system total flow rate.

EPS: The tolerance for interative solution of the pressure flow linear simultaneous equations

Numerical Solution for Pressure Equations

For each pass through the pressure-flow balance loop on each temperature iteration, a set of linear simultaneous equations are set up for each subnetwork. If the variable EPS is not supplied in the properties data, the solution to these equations is obtained using the Gauss-Jordan Elimination procedure. If the variable EPS is supplied, the equations are solved using Gauss-Siedel iterative method. The solution is then terminated when the change in pressure from one Gauss-Siedel iteration to the next is within EPS for all pressure nodes in the subsystem.

5.2.4.2 Flow Network and Subnetworks

The pressure/flow network is described by the flow network blocks and its corresponding subnetwork blocks for each flow system. The flow model consists of the pressure nodes, the flow tubes connecting the nodes and the fluid temperature nodes in each tube. (See sections 3.2.1 and 5.1.2 for descriptions of networks, subnetworks and model building methods). For the current version of the routine, the system network consists only of the inlet tube, the outlet tube and any other tubes in series with the inlet and outlet. The remainder of the system is contained in the subnetworks. Each subnetwork containes a set of interconnected tubes and are separated by network tubes.

The input requirements for each tube are the same whether the the tube is defined in the network or subnetwork. The input format is as follows for each tube:

NT, NP1, NP2 = (NF1, NF2 --- NFN)

Where NT = the tube number

NP1 = the pressure node on the upstream side of NT

NP2 = the pressure node on the down stream side of NT

NF1 --- NFN = all the fluid temperature node numbers contained in tube NT. If the user desires to control the conductor in the programming blocks, a blank or zero can be input between the ( ).

The heading card for the network data is,

'FLOW NETWORK'

The heading card for the subnetwork data is,

'SUBNETWORK = NSN'
GIN = NG
NSPR = NN

Where NSN = subnetwork number (must be unique)

NG = tube number for the tube carrying flow into the subnetwork

NN = the pressure node number at the subnetwork

5.2.4.3 Fluid Lump Data

The flow characteristics for each fluid temperature node contained in the system are supplied in the subordinate block headed by,

'FLUID LUMP DATA'

The following information is supplied for each fluid lump:

1. Fluid wetted perimeter
2. Fluid flow cross sectional area (perpendicular) to the direction of flow
3. Fluid Lump length
4. Method for calculating friction factor
5. Number of velocity head losses
6. Friction factor coefficient

The input format for the fluid lump data is:

NFL = WP, CSA, FLL, FFM, NKL, FFC

Where NFL = the fluid lump number (integer)
WP = the wetted perimeter for NFL (real)
CSA = the cross sectional area for NFL (real)
FLL = the fluid lump length for NFL (real)

FFM = the friction factor method (integer)
NKL = the number of head losses (integer or real)
FFC = the friction factor coefficient (real)

The following options are available for FFM, NKL and FFC:

If FFM = 0 or not supplied, the friction factor is calculated internally for the full range of Reynolds numbers using equations 50, 51, and 52.

If FFM ≠ 0, the friction factor is calculated internally for laminar flow using equation (50). FFM is then a curve number for a curve of friction factor VS Reynolds number which is interpolated at values of Reynolds numbers above 2000.

If NKL = 0 or not supplied, the head loss is 0.0

If NKL = a real constant, that is the value of the head loss

If NKL = a non-zero integer, it is then a curve number of a curve of head losses VS Reynolds number

If FFC = 0, or not supplied, FFC = 1.0

If FFC = a non-zero real number, then FFC = the real number.

FFC is applied to the friction factor in equation (49).

Several group input options are available for specifying the fluid lump numbers, NFL, on the left of the equal. These include:

(1) Random fluid lump numbers separated by commas:

NFL1, NFL2, - - - NFLN=WP, CSA, FLL, FFM, NKL, FFC

(2) A number of lumps separated by a constant increment:

NFL1 THRU NFLn BY INC = WP, CSA, FLL, FFM - - -

Where NFL1 is the starting fluid lump number of the group
NFLn is the ending fluid lump number of the group
INC is the increment between lump numbers. (If the increment is 1, the BY INC may be omitted)

(3) Any combination of (1) and (2) above. This includes multiple inputs of option (2) separated by commas.

5.2.4.4 Pump Data

The total flow rate entering each flow system is specified in the pump data which is headed by the heading:

'PUMP'

This block must always be supplied for each system. Four options are available for specifying the entering flow rate in the 'PUMP' block. These are:

(1) Constant entering flow rate
(2) Entering flow rate a function of time
(3) Flow rate a function of pump pressure rise as specified by a tabulated curve
(4) Pump pressure rise a function of flow rate as specified by a polynomial curve

A description of the input for these options is given below.

OPTION I : Constant Entering Flow Rate

The input format is:

NTl, LNS, VALUE

Where
NTl = An integer specifying the first (entering) tube in the system
LNS = An integer specifying the last (exiting) pressure node in the system
VALUE = A real number specifying the value of the flow rate

OPTION 2 : Entering Flow Rate A Function of Time

The input format is:

NTl, LNS, NCURVE

Where
NTL = An integer specifying the tube with entering flow
LNS = An integer specifying the last (exiting) pressure node in the system
NCURVE = An integer specifying a time dependent (Type 1) flow rate curve

OPTION 3 : Flow Rate A Function of System Pressure Rise Specified by Tabulated Curve

The input format is:

NTl, LNS, NCURVE

Where

NT1 = An integer specifying the tube with entering flow

LNS = An integer specifying the last (exiting) pressure node in the system

NCURVE = An integer specifying a curve number of a curve of flow rate vs pressure rise (Type 5 curve)

OPTION 4 : Pump Pressure Rise A Polynomial Function of Flow Rate

The input format is:

NT1, LNS, A0, A1, A2, A3, A4

Where

NT1 = An integer specifying the tube with entering flow

LNS = An integer specifying the last (existing) pressure node in the system

A0, A1, A2, A3, A4 = Curve fit constants for describing the pump curve

Using this option, the pump pressure rise is given by:

$$\Delta P = Ao+A1\ W + A2\ W^2 + A3\ W^3 + A4W^4$$

Where W = the flow rate

5.2.4.5 Valve Data

The input data required for each valve in a system is supplied in the valve subordinate block headed by;

'VALVES'

Three types of valves are currently available to the user. These types are identified by the following numbers:

Type 1 - Rate Limited Valve : The valve rate of movement is proportional to the sensed temperature error up to a maximum movement rate.

Type 2 - Polynomial Valve : The valve steady state position is a polynomial curve fit of the sensed error. A time constant may be input if a time lag is desired.

Type 3 - Switching Valve : The valve position is either the maximum position. On side one, the valve position is the maximum when the sensor temperature is above T1. Side two is the minimum when the sensor temperature drops below T2, side one position goes to the minimum and side two to the maximum.

Any of the above three types may be either one sided or two sided valves. Also, the set point may be either a constant value or a temperature lump number for Types 1 and 2. Thus, either types one or two may be used as (1) a single sided bypass valve for a cooling situation (radiator), (2) a single sided bypass valve for a heating situation (solar absorber for instance) or (3) a proportioning valve. Type 3 valves may be used as an on-off shut off valve, an off-on shut off valve, or a switching valve.

The general input format for each valve type consists of three integers on the left of an equal with 13 values on the right for rate limited, 17 values on the right for polynomial, and 11 values on the right for switching valves. The input values on the left are supplied in the following order:

1 - Valve Number: May be any unique integer number for identification.

2 - Valve Class Code: (See Section 3.2.1 and 3.2.3 for description of Class) Only Class 2 valves are available for the current routine configuration. Thus, the valve class must be the integer 2.

3 - Valve Type: 1 for bypass valve, 2 for polynomial valve, 3 for switching valve.

The input values on the right are described below for each of the three valve types.

Bypass Valve

The list of variables on the right of the equal must be supplied in the following order:

1 - Entering Tube Number : The tube number of the tube supplying the flow to the valve.

2 - Exiting Tube Number On Side 1 : The tube number of the tube leaving the valve on the side to be opened when the sensed temperature error becomes positive. If the valve is one sided and the characteristics of side 2 is desired (for instance, for a solar absorber bypass) this variable should be input as o.

3 - Exiting Tube Number on Side 2 : The tube number of the tube leaving the valve on the side to be opened when the sensed temperature error becomes negative. If the valve is one sided and the characteristics of the side 1 is desire (for instance, a radiator bypass) this variable should be set to 0.

4 - Initial Valve Position For Side One, X1 : (Should be between X1max and X1min.) The side 2 initial position is (1.0 - X1).

5 - Valve Operating Mode: If = 1, the valve operates normally if = 0, the valve does not operate but remains in its initial input position, X1.

6 - Minimum Position for Side One, X1min : Must be greater than 0.0. The maximum position for side 2 will be calculated by X2max = (1.0 -X1min). X1min. must be less than X1max.

7 - Maximum Position for Side One, X1max : Must be less than 1.0 and greater than X1min. The minimum value for side 2 will be calculated by (1.0-X1max)

8 - Sensor Lump For Side One or Set Point For Side 2 : If this variable is input as an integer it identifies the side one sensor lump to be controlled to (a) the set point or (b) sensor lump supplied for side 2 (next input). If this variable is input as a real number, it represents a set point to which the side 2 sensor lump will be controlled.

9 - Sensor Lump For Side 2 or Set Point For Side 1 : If this variable is input as an integer it identifies the side two sensor lump to be controlled to (a) the set point or (b) sensor lump temperature on side one (previous input). If this variable is a real number it represents a set point to which side 1 sensor lump will be controlled.

10 - Dead Band: The dead band as defined in Section 3.2.3.1 and Figure 4.

11 - Rate Factor: The valve rate factor as defined by Figure 4 and Section 3.2.3.1.

12 - Rate Limit : The valve rate limit as define by Figure 4 and Section 3.2.3.1.

13 - Geometric Factor : (For a valve class code of 2 only) The value of the variable E where ΔP valve = $E (\dot{W} 2/x^2)$ (See section 3.2.3.2)

Polynominal Valve

The list of variable to the right of the equal must be supplied in the following order:

1 - Entering Tube Number : The tube number of the tube supplying the flow to the valve.

2 - Exiting Tube Number On Side 1 : The tube number of the tube leaving the valve on the side to be opened when the sensed temperature error becomes positive. If the valve is one sided and the characteristics of side 2 is desired (for instance, for a solar absorber bypass) this variable should be input as o.

3 - Exiting Tube Number on Side 2 : The tube number of the tube leaving the valve on the side to be opened when the sensed temperature error becomes negative. If the valve is one sided and the characteristics of the side 1 is desirec (for instance, a radiator bypass) this variable should be set to 0.

4 - Initial Valve Position For Side One, X1 : (Should be between X1max and X1min.) The side 2 initial position is (1.0 - X1).

5 - Valve Operating Mode: If = 1, the valve operates normally if = 0, the valve does not operate but remains in its initial input position, X1.

6 - Minimum Position for Side One, X1min : Must be greater than 0.0. The maximum position for side 2 will be calculated by X2max = (1.0 -X1min). X1min. must be less than X1max.

7 - Maximum Position for Side One, X1max : Must be less than 1.0 and greater than X1min. The minimum value for side 2 will be calculated by (1.0-X1max)

8 - Sensor Lump For Side One or Set Point For Side 2 : If this variable is input as an integer it identifies the side one sensor lump to be controlled to (a) the set point or (b) sensor lump supplied for side 2 (next input). If this variable is input as a real number, it represents a set point to which the side 2 sensor lump will be controlled.

9 - Sensor Lump For Side 2 or Set Point For Side 1 : If this variable is input as an integer it identifies the side two sensor lump to be controlled to (a) the set point or (b) sensor lump temperature on side one (previous input). If this variable is a real number it represents a set point to which side 1 sensor lump will be controlled.

10 thru 15 - Polynomial Curve Fit Constants Ao, A1, A2, A3, A4, A5, : The steady state valve position on side one is given by (See Section 3.2.3.1)

$$X1_{ss} = Ao + A1\Delta T + A2\ (\Delta T)^2 + A3\Delta T^3 + A4\Delta T^4 + A5\Delta T^5$$

16 - Valve Time Constant : The polynomial valve time constant as described in Section 3.2.3.1. If a valve is desired with no time lag, a time constant which is small compared to the problem time increment should be input (Must be greater than zero)

17 - Geometric Factor : (For a class code of 2 only) The value of the variable E where

$$\Delta P_{valve} = E\ \frac{W^2}{X^2}$$

See Section 3.2.3.2

Switching Valve

The list of variables to the right of the equal must be supplied in the following order:

1 - Entering Tube Number : The tube number of the tube supplying the flow to the valve.

2 - Exiting Tube Number On Side 1 : The tube number of the tube leaving the valve on the side to be opened when the sensed temperature error becomes positive. If the valve is one sided and the characteristics of side 2 is desired (for instance, for a solar absorber bypass) this variable should be input as o.

3 - Exiting Tube Number on Side 2 : The tube number of the tube leaving the valve on the side to be opened when the sensed temperature error becomes negative. If the valve is one sided and the characteristics of the side 1 is desired (for instance, a radiator bypass) this variable should be set to 0.

4 - Initial Valve Position For Side One, X1 : (Should be between X1max and X1min.) The side 2 initial position is (1.0 - X1).

5 - Valve Operating Mode: If = 1, the valve operates normally if = 0, the valve does not operate but remains in its initial input position, X1.

6 - Minimum Position for Side One, X1min : Must be greater than 0.0. The maximum position for side 2 will be calculated by X2max = (1.0 -X1min). X1min. must be less than X1max.

7 - Maximum Position for Side One, X1max : Must be less than 1.0 and greater than X1min. The minimum value for side 2 will be calculated by (1.0-X1max)

8 - Valve sensor lump

9 - T1, Side 1 off temperature, Side 2 on temperature.

10 - T2, Side 1 on temperature, Side 2 off temperature.

11 - Geometric Factor : (For a valve class code 2 only) The valve of the variable E where

$$\Delta P_{valve} = E \frac{\dot{W}^2}{X^2}$$

See Section 3.2.3.2

5.2.5 Curve Data

The tabulated curves and tables required by the input data blocks and user subroutines are supplied in the data block headed by:

$ CURVES

Curves may be supplied in any order in this block. Curves are identified by a curve number which contains within it a code identifying its independent variable. The last digit of each curve number is used to identify the curve type, and thus, a curve number must contain at least two digits.

The following types of curves are currently available.

Curve Type	Description
0	Any array of numbers real or integers which the user may need for the programming blocks or user subroutine.
1	A doublet array with time as the independent variable.
2	A doublet array with temperature as the independent variable.
3	A doublet array with flow rate as the independent variable.
4	A doublet array with Reynolds number as the independent variable.
5	A doublet array with pressure drop as the independent variable.

With these codes, the curve numbers 10, 20, 30, 100, 1000, etc., are simple arrays for the user programming. The curve numbers 11, 21, 31, 41, 101, 151, etc., are used to specify time dependent tabulated functions which are referred to in the $ NETWORK DATA or $ FLOW SYSTEMS blocks. The numbers 12, 22, 32 are temperature dependent curves, 13, 23, 33 are flow rate dependent curves, etc.

The input format for the curves consists of (1) supplying the integer curve number and (2) supplying the curve values starting on the card following that with the curve number. The curve values are input for field separated by commas. If more than one card is required to supply the curve values, a continuation code (any non-zero character) must be supplied in column 75 for each additional card added.

A user may set up any number of locations in a curve for storing data by supplying an S followed by the number of spaces desired. For instance the entry

```
10/S, 1000
```

would set up curve number 10 with 1000 blank spaces. This option may be used anywhere within a curve. For instance,

```
20/ 1., 3., S, 500, 2.0
```

would set up a curve with 1., 3., 500 blank spaces and 2.0. The user should not specify zero values for the S option.

5.2.6 User Programming Blocks

5.2.6.1 General Description

Four input blocks are available to the user for supplying problem logic and calling on available user subroutines. In these blocks the user may perform logical operations on the temperature network elements and, to a lesser extent, on the pressure network elements. The user also has access to the time parameters of the problem, the curves and 75 constant locations which may be used either as real numbers or integers.

The four blocks differ only in their heading title and the point in the problem sequencing that the logic of the block is called upon. The block titles are given below with a description of their location in the problem sequence.

BLOCK TITLE	SEQUENCING
$ CENTRAL	Logical operations performed only one time for the problem. Specifies the call to the temperature solution subroutine and any user calls preceeding and following it.
$ PRETEMP	Logical operations performed prior to the temperature calculations on each iteration.
$ POSTTEMP	Logical operations performed following the temperature calculations and prior to the pressure/Flow calculations on each iteration.
$ OUTPUT	Logical operations to be performed on the print internal, WINC.

The location of the four user supplied logic blocks in relation to the overall problem flow is shown in Figure 9 .

The input format for the user programming blocks is simply the FORTRAN V Computer programming language and thus, valid FORTRAN V statement may be used. The first 5 columns are reserved for statements numbers or C in column 1 for comments. The * for comments cannot be used in the programming blocks. Per the FORTRAN language, the statements occur in columns 7 thru 72. Column 6 is used for continuation. User supplied statements for each block are combined with computer generated common and equivalence statements to form the following subroutines:

PROGRAMMING BLOCK	SUBROUTINE
CENTRAL	STEP 2
PRETEMP	PRETMP
POSTTEMP	POSTMP
OUTPUT	OUTPUT

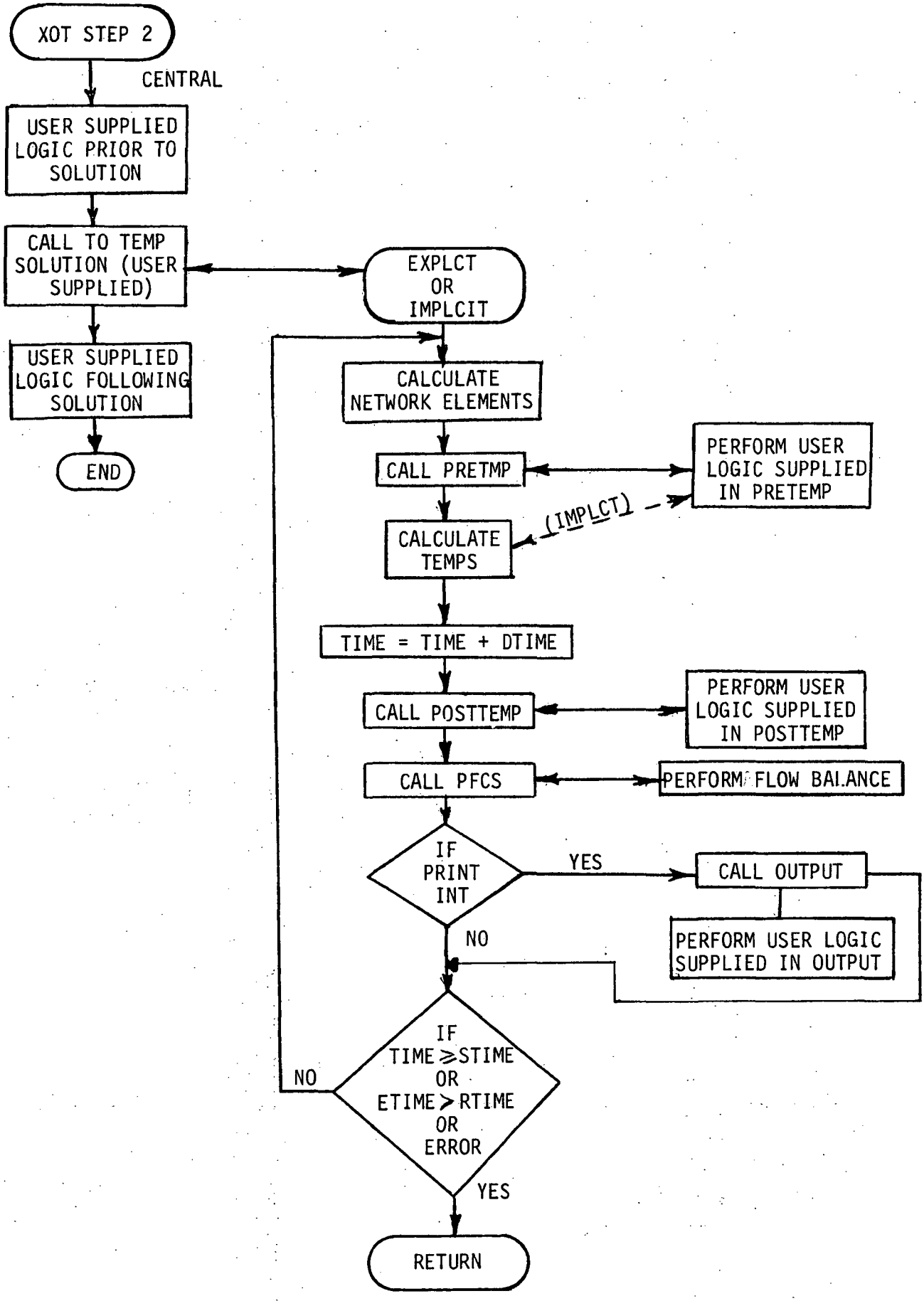


FIGURE 9 - OVERALL STEP 2 FLOW FOR MOTAR

The user has access to a number of problem variables in each of the programming blocks. These variables are in Labled common for each of the blocks and thus a change in one block to the problem variables is communicated to all the other blocks and to the problems. Variables available include the temperature network elements, pressure network elements, problem time realted variables, and user constants which may be used as integers or real numbers.

A list and description for the variables which may be referenced in any of the user programming blocks is given below:

VARIABLE	DESCRIPTION
T(I)	Temperature of temperature Node I
C(I)	Capacitance of temperature Node I
Q(I)	Heat Source for temperature Node I
U(I)	Conductance for conductor I
W(I)	Flow Rate for tube I
PR(I)	Pressure for pressure node I
G(I)	Pressure conductance for tube I
CURV(I)	Array contain all curve data
TIME	Problem time
STIME	Problem stop time
RTIME	Requested computer time (minutes)
WINC	Print interval
PINC	Plot interval for automatic plotting
C1 or K1 thru C75 or K75	Seventy-five (75) user variables which are communicated between the user subroutines. If used as a real number use the name CI. When used as an integer use the name KI. (CI and KI are equivalenced)

Any curve in the $ CURVES data may be addressed in any of the user blocks by use of the function LUTAB(I) where I is the curve number. The curve values are then obtained by addressing the array CURVE (J) where J = LUTAB(I). For example, if the fourth value of curve number 20 is desired, we would find it with the following two FORTRAN cards:

```
J = LUTAB (20)

C1 = CURV (J+4)
```

C1 will be assigned the values of the fourth location of curve number 20. This is the third data value, however, because the first location of each doublet table contains the number of points on the curve and the first location of a type zero curve contains the number of values. One must be careful when obtaining an integer value from a curve. Real variables should always be used when addressing curve since CURVE is real. However, the addressing variable, such as C1 above should be equivalenced to an integer variable name for use as an integer. If the fourth value of curve 20 were an integer in the example above, the variable name K1 would be used following the two statements since C1 and K1 are equivalenced.

The user may address the thermal network elements by addressing the arrays T, C, Q and U. The subscripts of T, C, and Q are the node numbers and the subscript of U is the conductor number. The $ PRETEMP block is located such that a definition by the user for any of the above variables overrides the computer definition prior to an iteration. A definition of these variables in any of the other blocks will be overriden by computer calculations however, except for (1) boundary temperatures, (2) constant capacitances and, (3) constant conductances. The Q values are set to zero prior to each iteration and thus a definition of Q in any block except $ PRETMP or by the + ABSORBED HEAT block will be lost. The values of Q supplied in $ PRETEMP will be added to values calculated by options in + ABSORBED HEAT.

The user has limited access to three elements of the flow network in the programming blocks. Available are overall tube conductance G(I), tube flow rate, W(I), and node pressure PR(I). Only the value of G(I) may be controlled by the user and this may be performed only if the tube lump number supplied for the tube is zero (with only one tube lump). Then the user must specify the value of the conductance. The call to the operations of the $ POSTTMP block has been placed immediately prior to the call to PFCS, the pressure flow analysis subroutine, so that the user may modify the G's as desired. The other variables, PR and W may be used for sensing purposes or independent variables only.

5.2.6.2 User Subroutines

Numerous subroutines have been developed and assembled in MOTAR for direct application by the user in the User Programming blocks. These subroutines greatly extend the MOTAR capability giving the user immediate access to numerous temperature solution subroutines, application subroutines, mathematical analysis and solution subroutines, interpolation subroutines, Matrix analysis subroutines, and output subroutines. A large number of these subroutines were taken directly from the SINDA computer routine[14,15]. This includes most of the mathematical analysis and solution subroutines, interpolation subroutines, and Matrix analysis subroutines. All of the temperature solution subroutines, many of the application sub-

routines (enclosure radiation, cabin analysis, heat exchangers, inline heaters, etc.) and some of the output subroutines were concieved during MOTAR development. A brief description of the available user subroutines is given below for each category of subroutines. A detailed description of all the subroutines is given in Appendix A.

Temperature Solution Subroutines

Temperature solution subroutines are generated by the computer during the pre-processing phase based upon the requirements of the input data. If the $ CENTRAL block is not included in the input (ie, no logic in this block) the computer will also generate a call to the appropriate transient temperature solution subroutine depending on the MPLCT code in columns 6 to 10 on parameter card 1. (If the code is 0, EXPLCT will be called, if 1, IMPLCT will be called). This call will be located in the processing phase main routine, STEP 2. If any logic appears in the $ CENTRAL block, the user must supply the call to the temperature solution subroutine in $ CENTRAL.

There are four potential temperature solution subroutines that may be specified by the user. The subroutines will actually be different from problem to problem since they are generated depending on the problem content. The temperature solution subroutines are:

1. EXPLCT - Performs the transient solution using the explicit method (two time increment options are available)

2. EXPSS - Performs the steady state solution for data stored in the explicit format

3. IMPLCT - Performs transient solution using the implicit method

4. IMPSS - Performs the steady state solution for data stored in the implicit format

EXPLCT and/or EXPSS must be used when the MPLCT code on parameter card 1 is a value of 0. IMPLCT and/or IMPSS routines must be used when the MPLCT code is 1. If the MPLCT code and the subroutine calls are inconsistent, the called subroutine will not be found and the problem will not be run.

The user may perform any number of calls to the transient or steady state subroutines or calls to both may exist in the same problem. For example, suppose a user desired to perform a parametric run in which the steady state solution is desired at TIME = 0. and a transient solution is desired starting at that condition for five different values of temperature for node ten. The logic in $ CENTRAL when using explicit method of solution would be:

```
      $ CENTRAL

       DO 20 I = 1,5
       TIME = 0.
       CALL EXPSS
       CALL EXPLCT
   20 T(10) = T(10) + 25.
```

If using the implicit method of solution the calls would be

```
      $ CENTRAL

       DO 20 I = 1,5
       TIME = 0.
       CALL IMPSS
       CALL IMPLCT
   20 T(10) = T(10) + 25.
```

With this type of programming a large number of problems may be performed with a single run.

A more detailed description of the temperature solution subroutines is given in Appendix A .

Application Subroutines

Several subroutines are available to the user which provide capabilities for specific applications. These are listed below and described in detail in Appendix A .

RADIR - Calculates the script-F values for IR radiation within an enclosure and uses these values to obtain the heat transfer during the problem. Permits consolidation of several temperature nodes on a single surface.

RADSOL - Calculates the script F values for radiation from an extended source entering an enclosure and uses these values to obtain heat transfer during the problem. Permits consolidation of several temperature nodes on a single surface.

IRRADI, IRRADE - Calculated the script-F values for IR radiation within an enclosure and uses these values to obtain the heat transfer during the problem.

SLRADI, SLRADE – Calcuates the script-F values for radiation from an entering source external to the enclosure and uses these values to determine heat transfer during the problem.

EFFEMS - Calculates the script-F between parallel flat plates.

SCRPFA - Calculates the script-F values for infrared radiation within an enclosure.

HXEFF, HXCNT, HXCROS, HXPAR – Simulates heat exchanger performance under steady state conditions - Can be used to approximate transient conditions.

HTRINL - Simulates a fluid inline heater.

CABIN - Performs a heat and mass balance on a cabin gas considering any number of entering streams and condensation on the cabin wall.

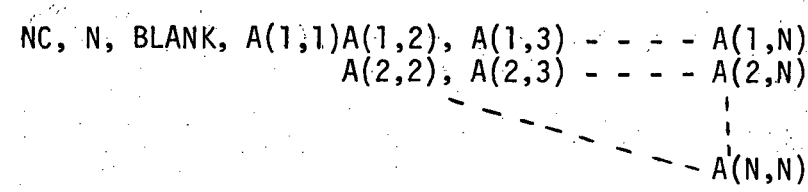
ABLATS - Represents a simple ablation (sublimation) capability

LQSLTR - Accounts for the phase change energy of a melting or solidifying material

LQDVAP - Allows the user to simulate the addition of liquid to a node

Matrix Analysis Subroutines

Numerious Matrix subroutines are available to the user for performing computations and manipulation on matrices of numbers. The matrices are stored in the $ CURVES block and must be in the proper format. For subroutines assuming rectangular matrices this format consists of (1) an integer specifying the number of rows in the first array data value, (2) an integer specifying the number of columns in the second array data value and (3) the Matrix values in row order. For subroutines designed specifically for symetric matrices the matrices are assumed to be stored in row order with each row starting with the diagonal value. The first value contains the size of the square Matrix and the second value isn't used. The format is,



A list of the matrix analysis subroutines available is given below. A detailed description of their use is given in Appendix A .

a) Special Matrix Generation

ZERO Generates a matrix such that every element is zero

ONES Generates a matrix such that every elements if one

UNITY Generates a square matrix such that the principal diagonal elements are unity and the remaining elements are zero.

SIGMA Generates a square matrix such that all elements on and below the principal diagonal are unity and the remaining elements are zero.

GENALP Generates a matrix such that every element is equal to a constant

GENCOL Generates a column matrix such that the first element is equal to X1 and the last element is equal to X2

FULSYM Forms a half symmetric matrix from a full square matrix

SYMFUL Forms a full square matrix from a half symmetric matrix

SYMFRC Forces symmetry upon a square matrix

DIAG Forms a full square matrix given a column or row matrix

UNDIAG Forms a row matrix from the diagonal elements of a square matrix

DIAGAD Adds the elements of a row matrix to the diagonal elements of a square matrix

b) Elemental Operations

ELEADD Adds corresponding elements of two matrices A and B to form a third Z (Matrix addition)

ELESUB Subtracts the corresponding elements of two matrices to form a third Z (Matrix subtraction)

ELEMUL Multiplies the corresponding elements of two matrices A and B to form a third Z

ELEDIV Divides the corresponding elements of two A and B matrices to form a third Z

ELEINV Obtains the reciprocal of each element of matrix A and place it in the corresponding location of another matrix Z

EFSIN Generates the sine of each elements of matrix A and places it in the corresponding location of another matrix Z

EFASN Generates the arcsine of each element of matrix A and places it in the corresponding location of another matrix Z

EFCOS Generates the cosine of each element of matrix A and places it in the corresponding location of another matrix Z

EFACS Generates the arcosine of each element of matrix (A) and places it in the corresponding location of another matrix (Z)

EFTAN Generates the tangent of each element of matrix (A) and places it in the corresponding location of another matrix (Z)

EFATN Generates the arctangent of each element of matrix (A) and places it in the corresponding location of another matrix (Z)

EFABS Generates the absolute value of each matrix (A) element

EFLOG Generates the natural log of each (A) element

EFSQR Generates the square root of each matrix (A) element

EFEXP Generates the exponential of each matrix (A) element

EFPOW Generates the power of each matrix (A) element

ADDALP Adds a constant to every element in a matrix

ALPHAA Multiplies every element in a matrix by a constant

MATRIX Allows a constant to replace a specific matrix element

SCALAR Allows a specific matrix element to be placed into a constant location

MATADD Adds a constant to a specific matrix element

c) Matrix Operations/Solutions

INVRSE Inverts a square matrix

MULT Multiplies two conformable matrices

TRANS Forms the transpose (Z) from matrix (A)

AABB Sums two scaled matrices

BTAB Performs the matrix operation $(B)^t (A)(B)$

BABT Performs the matrix operation $(B)(A)(B)^t$

DISAS Allows a user to operate on matrices in a partitioned manner by disassembling a submatrix (Z) from a parent matrix (A)

ASSMBL Allows a user to operate on matrices in a partitioned manner by assembling a submatrix (Z) into a parent matrix (A)

COLMLT
ROWMLT — Multiplies each element in a column or row of matrix (A) by its corresponding element from the diagonal matrix (V) which is stored as a vector

SHIFT — Moves an entire matrix as is from one location to another

REFLCT — Moves an entire matrix with the order of the column elements reversed from one location to another

SHUFL — Allows the user to reorder the size of a matrix as long as the total number of elements remains unchanged

COLMAX
COLMIN — Searches an input matrix to obtain the maximum or minimum values within each column

SYMREM
SYMREP — Allows the SINDA user to operate on a simple row/column of a half symmetric matrix

SYMDAD — Adds the elements of a vector array to the corresponding elements of the main diagonal of a half symmetric matrix

SYMINV — Obtains the inverse of a half symmetric matrix

POLMLT — Multiplies a given number of nth order polynomial coefficients by a similar number of mth order polynomial coefficients

POLVAL — Evaluates the polynomial for the input complex number X + iV, given a set of polynomial coefficients

PLYEVL — Evaluates each polynomial for each X value, given a matrix with nth order polynomial coefficients and a column matrix of X values

POLSOV — Calculates the complex roots, given a set of polynomial coefficients as the first row in a matrix

JACOBI — Determines the eigenvalues and eigenvector associated with an input matrix (A)

d) Store and Recall

CALL — Retrieves matrices on magnetic tapes

FILE — Stores matrices on magnetic tapes

ENDMOP — Used in conjunction with subroutines CALL and FILE. Causes all matrices from the logical 19 tape to be updated onto the logical 18 tape

LSTAPE Will output the name, problem number and size of every matrix stored on tape on logical 18.

e) Applications

MODES Solves a particular matrix dynamic vibration equation

MASS Generates an inertia matrix of a dynamic vibration system described in terms of deflections and rotations

STIFF Generates a stiffness matrix for a dynamic vibration system described in terms of deflections and rotations.

Interpolation Subroutines

Numerious interpolation subroutines are available to the user. Included are the capability for (a) linear interpolation of function one, two and three independent variables (b) parabolic interpolation of functions of one or two variables, and (c) lagrangian interpolations up to 50th order for one independent variable including extrapolation. Also included are options to to take advantage of cyclic curves, point/slope interpolation and linear extrapolation options.

A summary of the interpolation subroutines is given below. They are discussed in detail in Appendix A.

a) Lagrangian Interpolation

LAGRAN Uses one doublet array. Performs interpolation for order up to 50.

LGRNDA Uses two singlet arrays. Performs interpolation for order up to 50.

b) Linear Interpolation - Single Variable

POL Used one doubled array - one format

DIDEG1 Uses one doublet array - another format

D1D1DA Uses two singlet arrays

D1D1WM Uses D1DEG1 and multiplies the interpolation by the Z value

D11MDA Uses D1D1DA and multiplies the interpolation by the Z value

D1MDG1 Uses the arithmetic mean of two input values as the independent variable; uses a doublet array

D1M1DA Sames as D1MDG1 except two singlet arrays are used

D1M1WM Uses D1MDG1 and multiplies the interpolation by the Z value

D1M1MD Uses D1M1DA and multiplies the interpolation by the Z value

D1DG11
D1D11M
D1D1M1 Performs interpolation on an array of X's to obtain an array of Y's

D11DAL
D11D1M
D11MD1 Identical to D1DG11, D1D11M and D1D1M1, except for the use of singlet arrays and call on D1D1DA

D11MD1
D11MWM
D11M1M These are indexed subroutines which use the arithmetic mean of two input values as the independent variable

c) Linear Interpolation - Two Single Variables

CVQ1HT
CVQ1WM Performs two single variable linear interpolations

d) Parabolic Interpolation - Single Variable

D1DEG2 Uses LAGRAN and a doublet array

D1D2DA Uses LGRNDA and two singlet arrays

D1D2WM Uses LAGRAN and multiplies the interpolation by the Z value

D12MDA Uses LGRNDA and multiplies the interpolation by the Z value

D1MDG2 Uses the arithmetic mean of two input values as independent variable; uses doublet array

D1M2DA Sames a D1MDG2 except two single arrays are used

D1M2WM Uses D1MDG2 and multiplies the interpolation by the Z Value

D1M2MD Uses D1M2DA and multiplies the interpolation by the Z value

e) Cyclical Interpolation Arrays

D11CYL
DA11CY Reduces core storage requirements and uses linear interpolation

D12CYL
DA12CY Identical to D11CYL and DA11CY except that parabolic interpolation is used

D11MCY
DA11MC Identical to D12CYL and DA12CY except that the interpolation is multiplied by the value in address Z

DA12CY
DA12MC Identical to D11MCY and DA11MC except that parabolic interpolation is used

f) Point Slope Interpolations

GSLOPE — Generates a slope array so that point slope interpolation can be used

PSINTR
PSNTWM — Point slope interpolation

g) Bivariate Interpolations

BVSPSA
BVSPDA — Uses an input Y argument to address a bivariate array

BVTRN1
BVTRN2 — Constructs a bivariate array of Y's versus X and Z from an input array of Z's versus X and Y

D2DEG1 — Performs bivariate linear interpolation

D2DEG2 — Performs bivariate parabolic interpolation

D2D1WM — Uses D2DEG1 and multiplies the interpolation by the W value

D2D2WM — Uses D2DEG2 and multiplies the interpolation by the W value

D2MXD1
D2MXD2 — Identical to D2DEG1 and D2DEG2 except that the arithmetic mean of two X values is used as the X independent variable

D2MX1M
D2MX2M — Identical to D2D1WM and D2D2WM except that the arithmetic mean of two X values is used as the X independent variable

h) Trivariate Interpolations

D3DEG1
D3D1WM — Performs trivariate linear interpolation

i) Linear Extrapolation

ITRATE — Linearly extrapolates a new guess on the basis of Zero error

Output Subroutines

A number of subroutines are available to the user which permit the generation of printed, plotted or tape output for various situations in addition to the standard printing and plotting options. These subroutine provide the capability for printing individual values arrays and matrices; for plotting; and for writting and reading from magnetic tapes. The available subroutines are summarized below and described in detail in Appendix .

a) Network Printout

TPRNT	Prints thermal node temperature
CPRNT	Prints thermal capacitances
QPRNT	Prints the nodal heat flow values
UPRNT	Prints thermal conductances
DTPRNT	Prints the time increments
COPRNT	Prints the thermal network capacitances, heat flow values, time increment and the conductances
WPRNT	Prints flow rates
PPRNT	Prints pressures
VPRNT	Prints valve positions

b) Floating Point

PRINT
PRINTL — Allows individual floating point numbers to be printed for reference temperature, capacitance, etc.

c) Array Printout

GENOUT — Allows the output of any array of integers, floating point, or both
- GENI — Prints out an array of integer
- GENR — Prints out an array of real numbers

PRINTA — Allows the user to printout an array of values five to the line

PRNTMA
PRNTMI — Allows the user to print up to 10 arrays in a column format

PUNCHA — Enables a user to punch out an array of data values in any desired format

d) Plot Package

PRNPLT — Prints out a plot on the line printer

PLOTX1
PLOTX2
PLOTL1
PLOTL2
PLOTX3
PLOTX4 — Call upon a large package of subroutines specifically for the SC-4060

e) Tape Input/Output

READ
WRITE — Enables the user to read and write arrays of data as binary information on magnetic tape

f) Matrix Output

LIST	Prints the elements of a matrix and identifies each by its row and column number
PUNCH	Punches out a matrix, size n*n, one column at a time in any desired format
SYMLST	Prints out and identifies the element values of a half symmetric matrix

g) Special

PNTABL	Provides output information for users of subroutine ABLATS

Mathematical Subroutines

Several user subroutines are provided for mathematical operations and solutions. Capabilities provided the user include integration, root determination of cubic and quartic equations, polynomial calculation, simultaneous equation solution, least squares curve fit, and complex number calculations. These subroutines are summarized below.

a) Area Integration

SMPINT TRPZD	Performs area integration by Simpson's rule and trapezoidal rule using equal increments
TRPZDA	Performs area integration by the trapezoidal rule with non uniform increments

b) Roots

NEWTRT NEWRT4	Utilizes Newton's method to obtain one root of a cubic or quartic equation

c) Polynomial/Simultaneous Linear Equations

PLYNML PLYARY PLYAWM	Calculates the value of the dependent variable for an Nth order polynomial
SIMEQN	Solves a set of linear equations (10 or less) by the factorized inverse method

d) Curve Fit/Temperature Derivative

LSTSQU	Performs a least squares curve fit to an arbitrary number of X, Y pairs to yield a polynomial equation of up to order 10

Complex Variable Subroutines

a) Multiplication Operation

CMPXMP CMPYI	Multiplies two complex numbers or the corresponding elements of arrays of complex numbers

b) Division Operation

CMPXDV CDIVI	Divides two complex numbers or the corresponding elements of complex numbers

c) Roots

CMPXSR CSQRI	Obtains the complex square root of a complex number or array of complex numbers

Array Data Handling Subroutines

The capability of the MOTAR user to manipulate data stored in the $CURVES data is enhansed by use of numerious array data manipulation subroutines. The subroutines available are listed below and described in detail in Appendix A.

(a) Addition Operation

ADDARY Adds the corresponding elements of two specified length arrays to form a third array

ARYADD Adds a constant value to every element in an array to form new array

SUMARY Sums an array of floating point values

(b) Subtraction Operation

SUBARY Subtracts the corresponding elements of one array from another to form a third array

ARYSUB Subtracts a constant value from every element in an array to form a new array

(c) Multiplication Operation

MPYARY Multiplies the corresponding elements of two arrays to form a third

ARYMPY Multiplies each element of an array by a constant value to form a new array

SCLDEP
SCLIND Multiplies the dependent or independent variables of a doublet type interpolation array

(d) Division Operation

DIVARY Divides the elements of one array into the corresponding elements of another array to produce a third array

ARYDIV Divides each element of an array by a constant value to produce a new array

ARYINV Inverts each element of an array in its own location

ARINDV Divides each element of an array into a constant value to form a new array

ADARIN Calculates one over the sum of inverses of an array of values

(e) Distribution of Array Data

SHFTV Shifts a sequence of data from one array to another

SHFTVR	Shifts a sequence of data from one array and places data in reverse order in another array
FLIP	Reverses an array in its own array location
GENARY	Generates an array of equally incremented ascending values
BLDARY	Builds an array from a variable number of arguments in the order listed
BRKARY BKARAD	Distributes values from within an array to a variable number of arguments in the order listed
STOARY ARYSTO	Places a value into or takes a value out of a specific array location
STFSEQ STFSQS	Stuffs a constant value into a specified length array or group of sequential locations
SLDARY SLDARD	Moves array data values back one or two positions and updates the last one or two values
STORMA	Constructs historical data arrays during a transient analysis

(f) Singlet/Doublet Array Generation

SPLIT	Separates a doublet array into two singlet arrays
JOIN	Combines two singlet arrays into a doublet array
SPREAD	Applied interpolation subroutine DIDIDA to two singlet arrays to obtain an array of dependent variables versus an array of independent variables

(g) Comparison Operation

MAXDAR MXDRAL	Obtains the absolute maximum difference between corresponding elements of two arrays of equal length N

5.3 SPECIAL INPUT/OUTPUT FEATURES

Described below are the features available on MOTAR to permit the utilization of the magnatic tape capability of the Univac 1108 computer on input and/or output. Described are the data on tape with edit capabilities, restarting from and generating a restart tape, generating a history tape, starting from a history tape, plotting from a history tape and using flux curves from magnetic tapes.

5.3.1 Data on Tape with Edit

The input data described in Section 5.2.3 thru 5.2.6 must be supplied to the computer in the form of punched cards on the original run.

However, the data may be stroed on magnetic tape for input to subsequent runs by use of the Data Tape with Edit feature which is available on MOTAR. This feature increases the convenience and effectiveness of the use since the handling of large decks are not required. Also, the reliability of the tape reader is much higher than that for the card reader.

The EDIT routine is called by parameter INDATA input incolumns 4 and 5 on parameter card 2. Possible inputs are:

(1) INDATA = 0, All data is supplied on cards.

(2) INDATA = 1, All data is supplied on cards and the card images are written on tape on unit B. (Should be specified as an output tape on job card)

(3) INDATA = 2, Use data input on tape on unit C with desired changes on cards to write a new data tape on Unit B. (C is input tape and B is output tape)

(4) INDATA = 3, Use the data read in from unit B without change. Parameter cards 1 and 2 are read in from cards. (B is input tape)

If INDATA =-2, or -3, the card images on Unit B are punched.

When INDATA = +2, the deck set-up consists of parameter cards 1 and 2, the EDIT control cards (described below), and the new data cards (with the same format as the cards being replaced).

The EDIT control cards, used only when INDATA has a value of ± 2 are:

COLUMN	FORMAT	NAME	DESCRIPTION
1	A1	ID	* in column 1 identifies the card as an EDIT control card
6-15	I10	K3	Card number of first card to be removed if K3 is positive and K4>0. If K3 is negitive K31 is the card number of the card for which a merge correction will be performed. If K4 is blank or zero, cards change cards between this card and the next EDIT Control card will be add immediately following card K3.
16-25	I10	K4	Card number of last card to be removed prior to inserting the change cards in the data. If K3 is negitive, K4 is ignored.

As mentioned above, when K3 is negative, the merge option of edit is used. With this option the change card submitted after (K3) will be read in 5 column fields. For each 5 column field on the change card that is blank, no change will occur to the same field on the original card K3 . If any characters occur in a 5 column field that field on the original card will be replaced with the characters on the change card for the merged card. A $ in the 5th column of a 5 column field will cause that field to be blanked on the new merged card.

5.3.2 Dump and Restart Option

MOTAR is set up so that the problem is dumped on the third file of Unit I when either (1) the requested computer time, RTIME, or (2) the problem stop time, STIME, is exceeded. If the user assigns an output tape on Unit I this unit will contain the dumped information (on the second file) and must be saved if the problem is to be restarted. If restarting is planned the user must also save the K tape which contains the complied computer generated program.

When restarting a problem the user supplies the I tape from a previous run on the problem to be restarted as an input L tape positioned to the proper file (usually the third file). Also, the K tape received previously on output is assigned as an input unit K on restart. The deck setup for restart is described in Section 5.4.1.

When more than one call is made to a temperature solution routine (EXPLCT or IMPLCT) the dump information for each call is contained on the odd file of Unit I starting with the third. That is, the first call will dump on the third file; the second call on the fifth file, etc. The even files contain history information discussed in Section 5.3.3.

5.3.3 History Tape Options

The history tape contains information to permit automatic plotting of problem temperatures, flow rates, and pressures and valve positions as a function of time. This tape is written to permit the user to obtain the analysis results in the convenient plotted form with a minimum of user effort. A second feature of the history tape is the ability of the user to start a new problem from any time point for which data is written start from history tape option) The format of the history tape, plotting from the history tape and starting from the history tape are described below.

5.3.3.1 History Tape Format

History records are always written on the even files of Unit I with one history file for each call to a transient temperature solution subroutine (EXPLCT or IMPLCT). Normally problems will have only one call and thus, the history records are written on the second file with the dump for restart (Section 5.3.2) written on the third file. If the history files are to be saved for future use, the user must assign and save an I output tape

in the control cards (See Section 5.4.1) when making the run.

The history tape file on Unit I contains a number of logical records equal to the number of history times plus two. The first record contains a title and an integer count of the number of items to be written on the history tape. The second thru the next-to-last records contain the information to written on the history tape with a record for each time point. The history write interval, PINC, is specified on parameter card 2, columns 41 thru 50. When this interval is zero, it is set to the stop time minus the initial time so that two time points are plotted (at start and end of the problem).

The format for the history tape is as follows:

Record No. 1

Title (Written internally) including date and time of run in 12A6 format, 0, 0, 0, 0, 0, 0, No. of pressure nodes, number of valve positions, 0, 0, 0, number of tubes, 0, 0, number of nodes.

Record No. 2

Initial problem time pressures, valve positions, flow rates, node temperatures

Record No. 3

Second history time, pressures, valve positions, fl
Flow rates, Node temperatures

Record No. N+1 (Where N = number of history time slices to be written)

Last history time, Pressures, Valve positions, Flow rates, Node temperatures

Record No. N+2

Same as last record except time is negative.

5.3.3.2 Plotting From History Tape

The data written on the history tape may be used to generate SC-4020 CRT plots of the pressures, valve positions, flow rates and/or node

temperatures versus time. To accomplish this the user must submit a separate run using the plot routine, PLOTA, with the history tape as an input. In addition the user may combine the points of two or more history tapes into one so that the results of several runs may be presented on one plot frame This may be performed using either the PLOTA routine or the MCOMB routine. The MCOMB routine will shift the time point in addition to combining tapes if desired. The user may also plot the results of two separate runs on the same frame for comparison by using the routine COMPAR. Care must be taken when using this routine since a linear interpolation is performed at the comparison times prior to plotting.

The input descriptions for the routines PLOTA, MCOMB, and COMPAR are described in Appendix C.

5.3.3.3 Starting From a History Tape

The user may use the history tape data to start a problem. When using this option the problem input data supplied either on cards or on a data tape (using options 5.3.1) is input along with a history tape containing the desired starting conditions. A non-zero input in columns 8 and 9 of parameter card 3 and a value for TMPTIM in 61 thru 70 of parameter card 2 will cause the temperatures and valve positions to be read from the history tape on unit H at the first time point on the tape following TMPTIM. These new values will replace those of the original input. When using this option, there must be a one-to-one correspondence between the temperature nodes, and values in the model and on the history tape. Otherwise, the user may modify the model at will making this a very useful option.

5.3.4 Flux Tape Option

Incident heat curves may be read from tape by putting a non-zero value in column 6 of parameter card 3.

Restrictions on this option are:

(1) The initial block of curve data must be input on cards or data

(2) Particular curves must have the same number of points on each block of data read in as were input on cards initially

(3) Each curve may have a different number of points

(4) The curve data must be read in the same order the curves are in the data deck

(5) The first point on each curve in each block of data must be the same as the last point on that curve in the previous block of data

(6) All incident heat curves must be in a single block by themselves.

The data is read from a binary tape which has the following format:

Record No. 1

First Read Time

Record No. 2

Number of points on Curve No. 1 (Integer), Integer 1, Curve 1 independent variables, Curve 1 dependent variables, Number of points on Curve 2, Integer 1, Curve 2 independent variables, Curve 2 dependent variables, etc. for all curves.

Record No. 3

Second Read Time

Record No. 4

Same as Record No. 2

Record No. 5

Same as Record No. 1 but for the third read time.

Etc. until all blocks of data are on tape.

The amount of data which can be read in from tape is unlimited. The amount of data which can be read in a given block is dependent upon the data space available in the computer. It is possible to restart a problem which reads incident heat curve data from tape. The tape rewinds when the program is restarted; however, when the program calls for more incident heat curve data, it searches for the proper program time before reading. To use this option the flux tape must first be generated by G. E. routine LTVFTP which writes a flux tape in the proper format for use in the LTV routines. Incident heat start-up cards are also generated by this routine.

The following operations should be performed when using the flux tape option:

(1) Set flux tape code (NFLXCD) to one in column 6 of parameter card 3.

(2) Input start-up incident heat curves for all incident heats that will be read from the flux tape. Also, label all such curves as curve type.

(3) Assign the incident heat flux tape as input tape E.

5.4 RUN SUBMISSION REQUIREMENTS

This section describes procedures required to submit a MOTAR run on NASA/MSC Univac 1108 Computer. Included are (1) the card deck setup requirements (2) methods for estimating the amount of computer run time and and page output requirements and (3) methods for estimating storage requirements to establish maximum problem size.

5.4.1 Deck Setup Requirements

The MOTAR input data described in section 5.2 must be combined with the required system control cards price to submitting a run on the NASA/MSC Univac 1108 computer. The required control cards differ depending upon the input/output options and the corresponding devices required for a given problem.

Two fundamentally different types of runs may be made with MOTAR. These are (1) the runs in which the input data is supplied on cards or an input data tape (startup runs), and (2) runs which use a previously generated compressed data tape, and program elements as input (restart runs).

The deck setup for runs with the input data supplied by cards or data tape is shown in Table VIII. Included in the list shown in Table VIII are all the I/O unit that can possibly be assigned which include units A, B, C, E, F, H, I, and K. All of these units, except A, are optional and thus would not all be required for most problems. All may be used if required however. A description of the I/O devices is given below.

A is the device (logical Unit 1) to which the basic program tape should be assigned. It is always an input tape

B is the device (logical Unit 2) to which the final data tape (after editing) is assigned. When INDATA, parameter card number 2, is 1 or 2, B is an output tape. When INDATA is 3 B is an input tape.

C is the device (logical Unit 3) to which the data tape to be edited is assigned. C is required only when INDATA=3 and is always an input tape when assigned.

E is the device (logical Unit 7) to which the flux tape is assigned when NFLXCD, parameter card 3 is $\neq 0$. E is always an input

F is reserved for making program edits to A

TABLE VIII : Deck Setup for Runs with Input Data on Cards or Data Tape

```
7 8Z _ RUN
7 8n _ MSG
7 8  _ PLT   (Required only if user plotting subroutines are used)
7 8  _ ASC, A = XXXX
7 8  _ ASG _B = DATA or XXXX
7 8  _ ASG_C =  XXXX
7 8  _ ASG_E =  XXXX
7 8  _ ASG_F =  PFC                    Optional Assign Cards
7 8  _ ASG_H =  XXXX
7 8  _ ASG_I =  DUMP
7 8  _ ASG_K =  PROG
                    or 7 8 ASG  K,L if not planning a restart in the future
7 8  _ ASG_L
7 8  _ XQT _ CUR
_ _ TRW _ A
_ _ IN _ A
_ _ TRI _ A
_ _ TOC
7 8 _ XQT _ STEP1
    (Parameter Card No. 1)
                                 )
                                 ) Input data deck or data edits
                                 )
7 8 _ FOR, K _ STEP2, STEP2
7 8 _ FOR, K _ TEMPTR, TEMPTR
7 8 _ FOR, K _ PRETMP, PRETMP
7 8 _ FOR, K _ POSTMP, POSTMP
7 8 _ FOR, K _ OUTPUT, OUTPUT
7 8 _ FOR, K _ TEMPSS, TEMPSS
                      )
                      ) User Subroutines
                      )
```

TABLE VIII CONTINUED

7_8 _ XQT _ CUR _ _ TRW _ K _ _ DUT _ K _ _ TEF _ K _ _ ERS _ _ TRW _ K _ _ IN _ K _ _ TRI _ K	Required only if restarting is planned on future runs
7_8 _ XQT _ STEP 2	
(PARAMETER CARD 2)	
(PARAMETER CARD 3)	
7_8 _ EOF	

H is the device (logical Unit 10) to which the history tape is assigned when NEWTMP≠0 . H is always an input tape.

I is the device (logical Unit 11) which is assigned for writing history records for future plotting and for writting the dump for restart. I is always an output tape.

K is the device (logic Unit 13) which will contain the basic program from Unit A, plus the programs generated during the preprocessing phase. K should be saved as an output tape if restarting is planned in the future. If no restarting is planned, K should be assigned to a Fostran file.

L is the device (logical Unit 14) which contains the compressed data for input to the processing phase. On a restart run, it is an input tape; on other runs it is a Fostran file.

In addition to the above, two FH 432 drum files - M and N are used during the preprocessing phase. Thus, the user should identify the fact that two such files are needed on the MSG card.

The deck setup for runs with the input data supplied by a compressed data tape or restart tape is shown in Table IX For this type of run, Units A and L are required and units E, F, H, and I are optional. The units are the same as described above.

5.4.2 Estimation of Computer Time and Output

The computer time required for a MOTAR run using the EXPLCT routine on the Univac 1108 may be estimated by the following relation:

$$CTIME = \frac{STIME - TIME}{DTIME}\left[\frac{(NODES)\ FC}{810{,}000}\ (1+2\ FVP) + 1.23 \times 10^{-5} + \frac{HFC}{306{,}000} + \frac{NCC}{75{,}600} + \frac{NFL}{104{,}400}\right]$$

Where CTIME = the required computer time in minutes

STIME = the problem stop time in the problem units

TIME = the initial problem time in the problem units

DTIME = the problem time increment

TABLE IX Deck Setup for Restart Runs

```
7 8z _ RUN
7 8n _ MSG
7 8 _ PLT
7 8 _ ASG _ A = XXXX
7 8 _ ASG _ E = XXXX
7 8 _ ASG _ F = PCF
7 8 _ ASG _ H = XXXX
7 8 _ ASG _ I = DUMP
7 8 _ ASG _ L = START
7 8 _ XQT _ CUR
_ _ PEF _ L/2
_ _ TRW _ A
_ _ IN _ A
_ _ TRI _ A
_ _ TOC
7 8 _ XQT _ STEP 2
   PARAMETER CARD 2
   PARAMETER CARD 3
7 8 _ EOF
```

Optional Control Cards

PTIME = the problem print interval

NODES = the number of nodes in the problem

FC = the sum of the conduction conductors plus the number of radiation conductors divided by number of NODES

FVP = the approximate fraction of capatance and conductors that contain variable properties

NFC = Number of flow conductors

NCC = number of convection conductors

NFL = number of fluid lumps

For the Univac 1106, CTIME should be multiplied by approximately 1.7. The time required for the IMPLCT solution routine is given by,

$$CTIME = \frac{STIME - TIME}{DTIME} \left[\frac{(NODES)\ FC}{150{,}000} (1+2\ FVP) + \frac{NFC}{85{,}000} + \frac{NCC}{21{,}000} + \frac{NFL}{104{,}400} \right]$$

The above estimates do not include any provisions for user programmer subroutine time. Additional time should be allowed for any such programs.

The number of page of printed output using the standard output options may be estimated as follows,

$$NP = \frac{NTUBES + NPN + NODES}{170} \quad \frac{STIME - TIME}{PTIME}$$

Additional pages should be allowed for any output subroutines or write statements specified by the user in addition to the standard.

5.4.3 Data Storage Requirements

The basic storage requirements for a thermal problem using the EXPLCT solution subroutine is given by:

$$NSPTH = 5*NODES+3*NTC+4*NCC+3*NFC+NSPCC+3*NTSPC+3*NAH+4*NCURY+SNPTS+ISANFS+7*(NSUBS+1)+(MXPNSS-1)^2+6*NTBS+21(MXPNSS-1)+10*NFLTPS+20*NVLVS+NPN+4*NFL$$

Where NSPTH = total thermal space required

NODES = number of nodes required

NTC = total number of conductors

NCC = number of convection conductor

NFC = = number of flow conductors

NSPCC = number of special option conduction conductors (not constant)

NTSPC = number of types of special capacitances (each group entry is a type)

NAH = number of specified absorbed heat values

NCURV = the total number of curves

SNPTS = the sum of the number of points

NFS = the number of flow systems

NSUBS = the number of subsystems

MXPNSS = the number of pressure nodes in the subsystem with the maximum number of pressure nodes

NTBS = the number of flow tubes

NFLTPS = the number of fluid lump types

NVLVS = the number of valves

NPN = the number of pressure nodes

NFL = the number of fluid lumps

The storage requirements for a thermal problem using the IMLCT method of solution is that given above plus 2*NTC.

5.5 OUTPUT DESCRIPTION

Two types of standard output are available with MOTAR. The first is a normal print at the input print. The second is a check out print

which occurs on every iteration when requested (When NCKOUT on parameter card 3 in non-zero). This checkout print consists of a normal print plus a considerable amount of addition information which is useful in checking out the input data deck. The checkout print formats for the EXPLCT and IMPLCT solution subroutines are different.

In addition to the two standard output options available on MOTAR, the user may specify any of a large number of user output subroutines these are not described in this section but a description may be found in Appendix A.

The normal MOTAR printing, the EXPLCT checkout printing, the EXPLCT checkout printing and the IMPLCT checkout printing are described below.

5.5.1. Normal Printing

Some examples of normal printing are given in the sample problems described in Appendix B. A explanation of the terms apperaring in the normal print is given below. The units are always the problems units.

The beginning of a print interval is indicated by the printout of the time parameters. This is followed in order by the flow rates per tube, the pressures per pressure node, the valve positions per valve and the temperatures per node. These are described below as they appear.

Time Parameters

MISSION TIME - The mission time in the problem units

COMPUTATION INTERVAL- The computation time increment in the problem units

COMPUTER TIME - The amount of computer time used to this point (since the XQT SETP2) in minutes

Flow Rates

Flow rates are printed in numerical order by tube in the problem units. Five flow rates are output on each line with the tube number of the fifth flow rate printed to the right of it.

Pressures

Pressures are printed in numberical order by pressure node in the problem units. Five pressures are printed out per line with the pressure node number of the fifth pressure printed out to the right of it.

Valve Positions

The valve positions are listed one to a line in order of valve number. Each is identified with its number.

Temperatures

Temperatures are printed out in numerical order of the node numbers. The output is the problem unit. Temperatures are printed out five to a line with the lump number of the last temperature on each line printed just to the right of it.

5.5.2 EXPLCT Checkout Printing

The checkout printing may be used to examine internally calculated values such as heat transfer coefficients, frictional and bend loss pressure drops, and maximum time increments. An examination of the time increments for each lump may reveal that certain lumps which are not thermally important to the solution have small time increments. In that case the time increment may be selected such that fewer iterations are required. Care should be exercised that the lumps with overridden time increments do not affect the transient analysis.

An explanation of the terms appearing in the checkout printing for the EXPLCT routine is given below:

CHECKOUT PRINT FOR PRESSURE-FLOW COMPUTATION SUBROUTINE

MPASS	-	Number of temperature iterations between pressure balance
MXPASS	-	Maximum number of passes to balance in PFCS
NSS	-	Number of Subsystems in this system
LOCI	-	Starting location in the subsystem data for the subsystems of this system
NVLOC	-	Starting location in the valve data for the valves of this system
LOCP	-	Starting location in the pump data for the pumps of this system
TOL	-	Subsystem solution tolerence
EPS	-	Relaxation tolerence for numerical equation solution
NTB	-	Tube number
W(NTB)	-	Flow rate in tube NTB
NFRM	-	Upstream pressure node for tube NTB
NTO	-	Downstream pressure node for tube NTB
NF	-	Number of fluid lumps in tube NTB
LN	-	Fluid lump number
T(LN)	-	Temperature of fluid lump number LN
LOC4	-	Location in the fluid lump data array for lump LN data
WP	-	Wetted peirmeter of lump number LN

CSA2 - Flow cross-sectional area squared for lump LN
MFF - Method code for determining friction for lump LN
HL - Number of head losses in fluid lump LN
~~FLL/D - Fluid lump length divided by diameter for lump LN~~
RO - Density of fluid lump LN
MU - Viscosity of fluid lump LN
RE - Reynolds number of fluid lump LN
FFC - Friction factor coefficient for fluid lump LN
FF - Friction factor for fluid lump LN
R(LN) - Flow resistance for fluid lump LN
PCHK - Pressure at the upstream node for tube NTB
KPASS - Number of passes thru the system
NPASS - Pass number for balancing subsystem
INLT - Inlet tube number for the subsystem
NPR - Specified pressure node for the subsystem
NT - Number of tubes in the subsystem
LOC2 - The starting location in the tube data for the tubes of this subsystem
LOC3 - The starting location in the fluid lump data for the fluid lumps in this subsystem
NP - Number of pressure nodes in the subsystem
NTU - Number of tubes upstream of subsystem
G(NTB) - Flow conductor for tube NTB

Time Parameters

MISSION TIME - The mission time in the problem units

COMPUTATION INTERVAL - The computer time increment in the problem units

COMPUTER TIME - The amount of computer time used at this point (since XQT STEP2) in minutes

Flow Rates

Flow rates are printed in numerical order by tube in the problem units. Five flow rates are printed per line with the number of the fifth being identified by an integer at the end of the line.

Pressures

Pressures are printed in numerical order by pressure node in the problem units. Five pressures are printed out per line with the pressure node number of the fifth pressure printed out to the right of it.

Valve Positions

The valve positions are listed one to a line in order of valve number. Each is identified with its number.

Temperatures

Temperatures are printed out in numerical order of the node number. The output is in the problem units. Temperatures are printed out five to a line with the lump number of the last temperature on each line printed to its right.

Capacitances

Capacitances are prined out in numerical order of the node number. The output is in the problem units. These are printed out five to a line with the lump number of the last capacitance on each line printed to its right.

Heat Storage Rates

Heat storage rates are printed out in numerical order of the node number. The output is in the problem units. These are printed out five to a line with the lump number of the last rate on each line printed to its right.

Conductances

Conductances are printed out in numerical order of the conductor number. The output is in the problem units. These are printed out five to a line with the conductor number of the last conductor on each line printed to its right.

Time Increments

Time increments which are used in the temperature calculation are printed out in numerical order of the node number. The output is in the problem time units. These are printed out five to a line with the lump number of the last time increment on each line printed to its right.

CHECKOUT PRINT FOR FLOW CONDUCTOR COMPUTATION SUBROUTINE

NC - Conductor number

NFL - Lump number of the "from" lump

NTL - Lump number of the "to" lump

T(NFL) - Temperature of the from lump

KODEF - Code specifying method for obtaining flow rate

CP - Specific heat

CHECKOUT PRINT FOR CONVECTION CONDUCTOR COMPUTATION SUBROUTINE

NC - Conductor number

NFL - Fluid lump number

NTBL - Tube lump number

NFS - Flow system number

NTB - Tube number

MHTC - Code for identifying method for heat transfer coefficient calculation

LOC5 - Location in the type data array for the flow data for this lump

AHT - Heat transfer area

F1 - Entry length laminar flow multiplying factor

F2 - Fully developed laminar flow multiplying factor

T(NFL) - Fluid lump temperature

CP - Specific heat

VIS - Viscosity

CON - Conductivity

W(NTB) - Flow rate in tube NTB

WP - Wetted perimeter

RE - Reynold's number

PR - Prandtl number

D - Hydraulic Diameter

H - Heat transfer coefficient

X/D - Distance from tube enterance divided by hydraulic diameter

U(NC) - Calculated conductor value

ST - Stantion number when MHTC = 3, otherwise, zero

5.5.3 IMPLCT Checkout Printing

The checkout printing for the IMPLCT routine is identical to that for the EXPLCT routine given in Section 5.5.2 with the additions described below.

CHECKOUT PRINT FOR IMPLICIT TEMPERATURE CALCULATION SUBROUTINE

NPASS	-	The pass number in obtaining the temperature solution
NODE	-	The temperature node number
T(NODE)	-	The temperature of NODE
RHS(NODE)	-	The right hand side for NODE as given by the right hand side of Equation 14
C(NODE)/TINC	-	The capacitance of NODE divided by the problem time increment
QSUM(NODE)	-	The value given by RHS(NODE) minus the second term on the left side of Equation 14
USUM(NODE)	-	The coefficient of T_i^{n+1} in Equation 14
DELTAT	-	The temperature change for the pass for NODE after applying the overrelaxation parameter
TNEW	-	The temperature of NODE following the pass

6.0 REFERENCES

1. Myers, D. E., Analytical Methods in Conduction Heat Transfer, McGraw-Hill, 1971

2. Nuttal, H., "A Note on a Classical Problem in the Mathematical Theory of Heat Conduction", Int. J. Engn Sci. 5(1), 39 (1967).

3. Reitzel, J., "Use of Boundary Sources in Problems of Heat Conduction", J. Appl. Phys. 38 (10), 3808 (1967).

4. Crosbie, A. L. and Viskanta, R., "A Simplified Method for Solving Transient Heat Conduction Problems with Non-linear Boundary Conditions", J. Heat Transfer 90(3), 358 (1968).

5. Polyakov, Yu. A., "Exponential Point Method in Nonsteady-State Heat Transfer", High Temperature 5(1), 117 (1967).

6. Haimo, D. R., "Series Representation of Generalized Temperature Functions", HASA CR-81746, Southern Illinois University, Edwardsville, Ill. (1966).

7. Lynch, J. H., "Heat Conductionaas an Equivalent Eigenvalue Problem", NASA TM X-1362 (1967).

8. Sparrow, E. M. and Hajo-Sheikh, A., "Transient and Steady Heat Conduction in Arbitrary Bodies With Arbitrary Boundary and Initial Conditions", J. heat Transfer 90(1), 103 (1968).

9. Schneider, P. J., Conduction Heat Transfer, Addison-Wesley, 1955.

10. Carslaw and Jagger, Head Conduction in Solids.

11. Leach, J. W., "Thermal Equilibrium Extrapolation Technique", VMSC Rept. 00.1083, 30 July 1968.

12. Gaddis, J. L. "Explicit Finite Difference Heat Transfer Program - LVVM25" LTV Report No. 00.823, 29 July 1966.

13. Dusinberre, G. M., Heat Transfer Calculations by Finite Difference, International Textbook Co., 1961.

14. Smith, J. P., "SINDA Users Manual", TRW Report 14690-H001-R0-00, April 1971.

15. Ishimoto, T. and Fink, L.C., "Systems Inprove Numerical Differening Analyzer Engineering Program Manual," TRW Report 14690-H002-R0-00, June 1971.

16. Hardi, P. D., Howell, H. R. , Williams, J. L. ,"Lunar Module Ascent Stage Thermal Simulator", LTV Report No. 350.3, 11 August 1957.

17. Eckert, E. R. G., and Drake, R. M., Heat and Mass Transfer, McGraw-Hill Book Company, Inc., New York, New York, 1959

18. Sparrow, Eckert, Jonsson, "An Enclosure Theory for Radiative Exchange Between Specularly and Diffusely Reflecting Surfaces, " Trans. ASME 84, 294, 1962

19. French, R. J., "Computer Program for Space Radiator Analysis and Design", VMSC Report No. 00.391, 25 February 1964.

20. Gaddis, J. L., "Implicit Finite Difference Generalized Heat Transfer Program"- LTV Routine number LVVM22, VMSC Report No. 00.809, 21 July 1966.

21. Hixon, C. W., Hardi, P. D. and Williams, J. L., "Apollo Block II Command Module Thermal Simulator" VMSC Report No. 350.2, 28 July 1967.

22. Oren, J. A., "Computer Program and Performance Predictions for Apollo Block II Power Generation and Heat Rejection System," VMSC Report 350.7, 29 September 1967

23. French, R. J. and Gaddis, J. L., "Computer Program for Apollo Fuel Cell Radiator-Condenser Cooling System Analysis", VMSC Report 00.704, 30 October 1965, revised 17 December 1968.

24. Emmons, H. T., Hardi, P. D., and Williams, J. L., "Lunar Module Descent Stage Thermal Simulator", Report No. 350.4, 18 August 1967.

25. Williams, J. L., "Apollo Block I Command Module Computer Program and Thermal Model", VMSC Report 00.808, 28 November 1966.

APPENDIX A

USER SUBROUTINES

1.0 INTRODUCTION

This appendix presents a description of all user subroutines currently available in MOTAR. Table A-1 presents the location of each of the seven categories of subroutines. Table A-2 presents an alphabetical listing of the user subroutines with the corresponding page numbers.

Table A-1

Table A-2

ALPHABETICAL LISTING

Table A-2
Alphabetical Listing (Continued)

2.0 SUBROUTINE DESCRIPTION

2.1 TEMPERATURE SOLUTION SUBROUTINES (Computer generated and vary from problem-to-problem)

SUBROUTINE NAME: EXPLCT

PURPOSE:

This subroutine obtains the transients temperatures using the explicit forward differencing solution method described in Section 3.1.2.1. The calculations for each node are given by equation (12) and the stability criteria is given by equation (13). Two options are available for application of the stability criteria. There are

(1) Override or Floating Option - (Specified by 0 in columns 11 through 20 of parameter card 2):

Temperature calculation is made for each lump using the input time increment or the maximum stable increment (equation 13), whichever is smaller.

(2) No Overriding Option - (Specified by a nonzero entry for TINCMN in columns 11 through 20 of parameter card 2):

Temperature calculations for all lumps are made using the input time increment or the smallest maximum stable increment whichever is smaller (all lumps use the same increment). A minimum value for the time increment is specified by the user and the problem is terminated if the maximum stable increment drops below the input minimum.

EXPLCT is generated during the preprocessing phase of MOTAR based upon the input data and thus will vary in actual form from problem to problem, depending upon the requirements. A functional flow chart is shown in Figure A-1, which shows the order of calculations including the calls to the user programming blocks. Many of the operations shown will not be present when the data doesn't require them.

RESTRICTIONS:

Must be called from the $CENTRAL block. MPLCT code in columns 6 to 10 of parameter card 1 must be 0 or blank.

CALLING SEQUENCE: EXPLCT

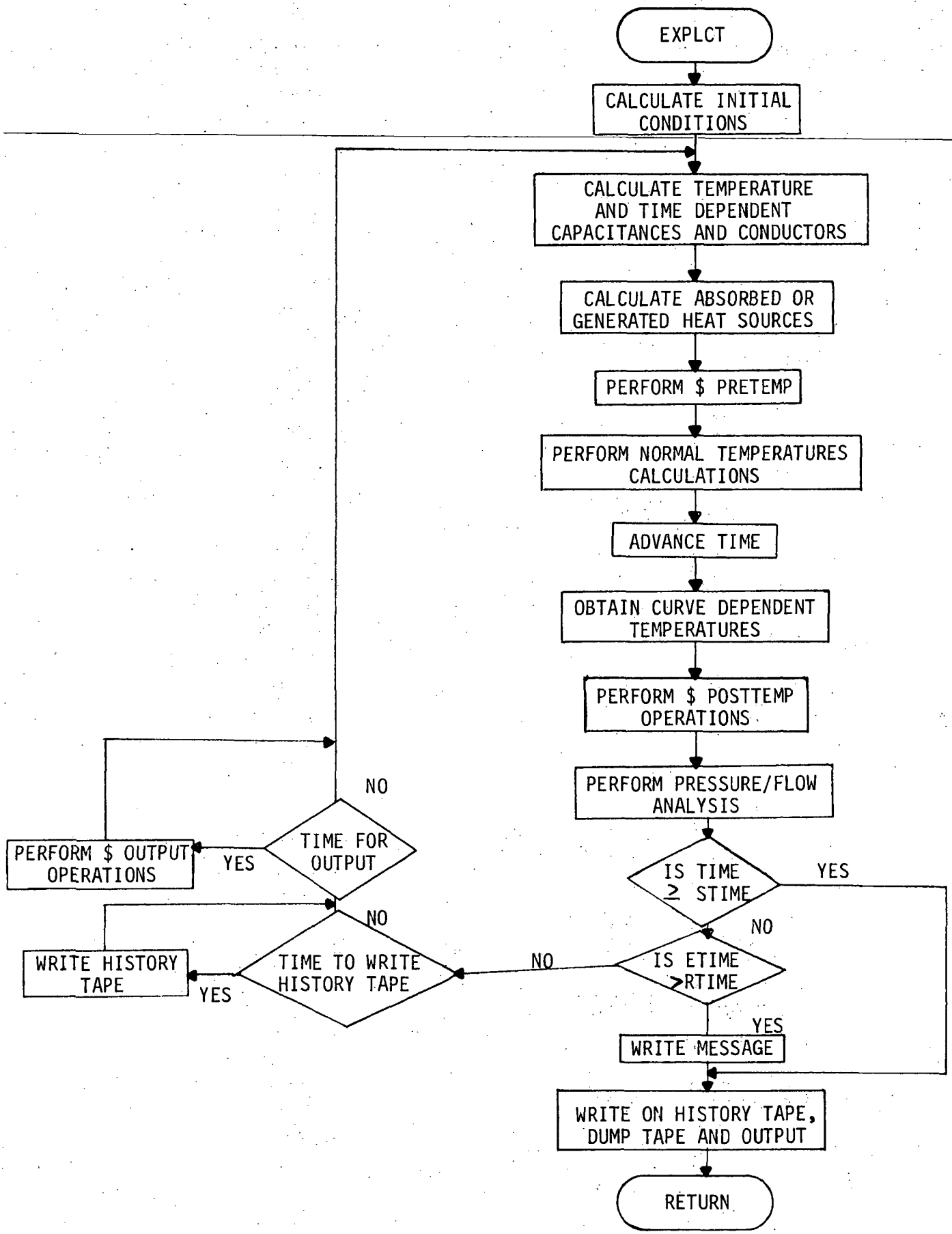


FIGURE A-1: FUNCTIONAL FLOW CHART OF SUBROUTINE EXPLCT

SUBROUTINE NAME: EXPSS

PURPOSE:

This subroutine obtains the steady state temperature distribution for problems with the data stored in the explicit format. It is compatible with the EXPLCT subroutine and, thus, may be used on the same problem. It iterates the basic transient explicit temperature equation (Equation 12) to a solution while holding time constant. Convergence is accelerated by applying the maximum stable time increment given by Equation 13 to each node. This method results in a block iterative solution method. Compared with subroutine IMPSS, the convergence is slower but less storage space for the data is required.

A functional flow chart for EXPSS is shown in Figure A-2.

RESTRICTIONS:

MPLCT code in columns 6 thru 10 of parameter card 1 must be 0. Must be called from $CENTRAL

CALLING SEQUENCE: EXPSS

SUBROUTINE NAME: IMPLCT

PURPOSE:

This subroutine calculates the transient temperature distribution using the implicit method of solution discussed in Section 3.1.2.2. Either backward difference, mid-difference or any point in between may be specified by the value of ALPHA in columns 16 thru 20 of parameter card 3, which is constrained to be between 0.5 and 1.0. When ALPHA is 1.0 the solution is backward-difference; when ALPHA is 0.5 the solution is mid-difference. A modified version of the successive-point-over relaxation iteration method is used for equation solution. The over-relaxation parameter, ORP, is supplied in columns 26 thru 30, and the solution tolerence DTMXA is supplied in columns 21 thru 25, both on parameter card 3.

Figures A-3 and A-4 provide functional flow charts of IMPLCT and TEMPI which is called from IMPLCT. As may be seen, the $PRETEMP operations are preformed during the relaxation loop of TEMPI, and thus, the user calls in $PRETEMP are considered during relaxation. The user may not reference the C array from $PRETEMP when using IMPLCT but may perform operations on the Q, U, and T arrays. C operations may be performed in the $POSTTEMP block.

RESTRICTIONS:

MPLCT code in column 10 of parameter card 1 must be 1. IMPLCT must be called from $CENTRAL. The C array is not available to the user in the $PRETEMP operations block.

CALLING SEQUENCE: IMPLCT

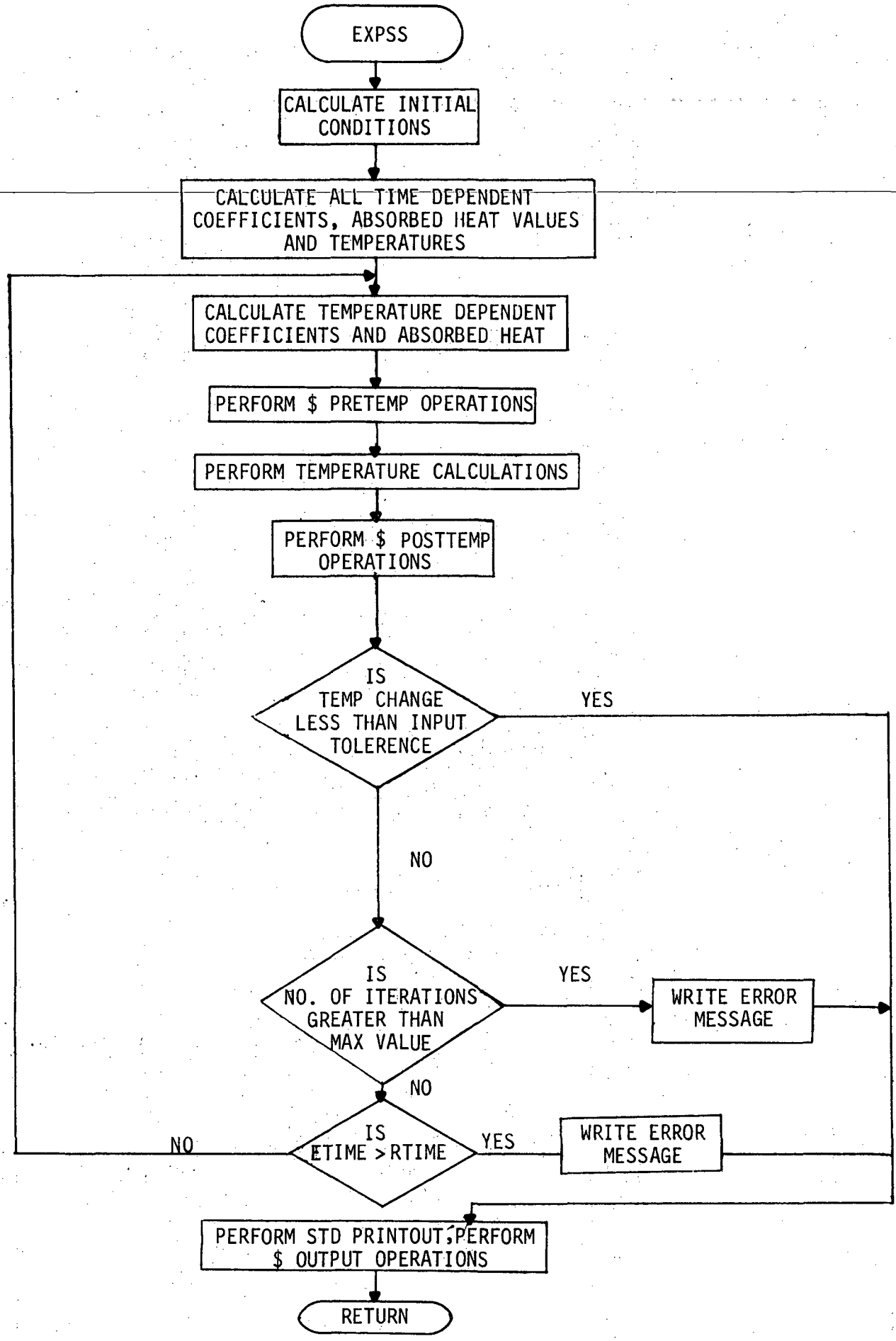


FIGURE A-2: FUNCTIONAL FLOW CHART OF SUBROUTINE EXPSS

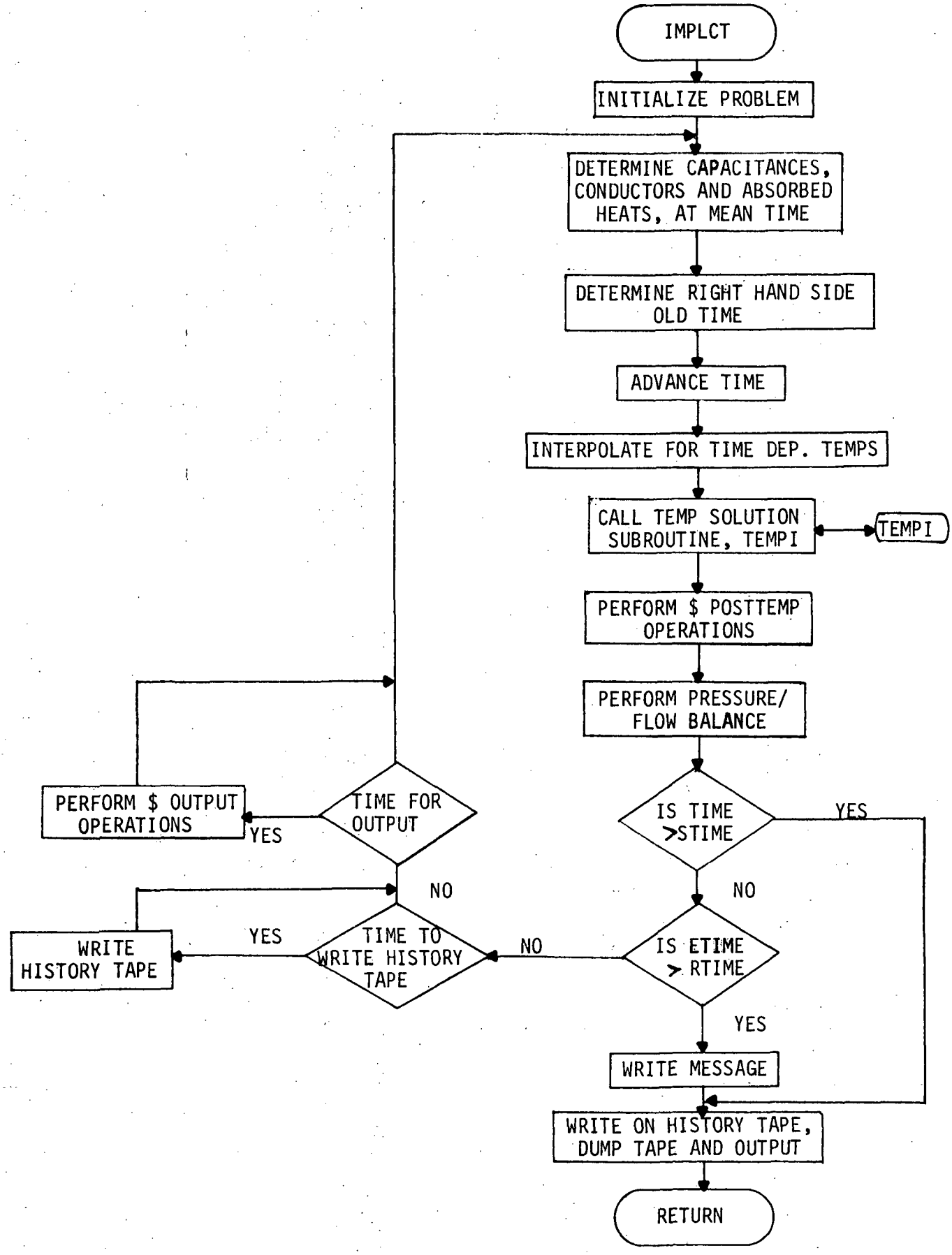


FIGURE A-3: FUNCTIONAL FLOW CHART OF IMPLCT

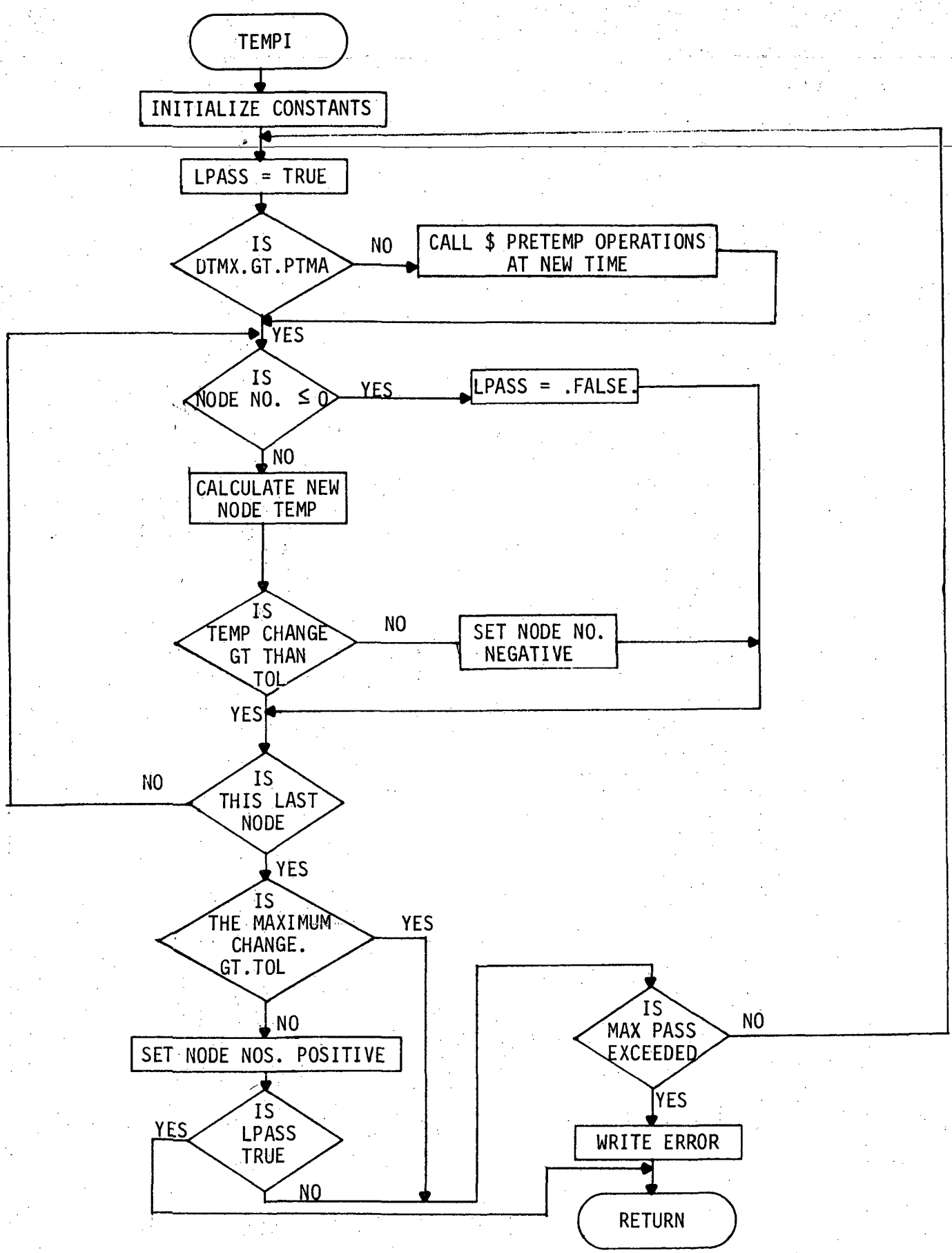


FIGURE A-4: FUNCTIONAL FLOW CHART OF TEMPI

SUBROUTINE NAME: IMPLSS

PURPOSE:

This subroutine calculates the steady state temperature distribution for problems with the data stored in the implicit format. It is compatible with IMPLCT in that data stored for either may be used with the other. Thus, IMPLCT and IMPLSS may be used on the same problem.

Subroutine IMLSS performs the same basic iteration procedure performed in IMPLCT except the time is held constant and the capacitance is assumed to be zero. Functional flow charts of IMPLSS and TEMPSS which it calls are shown in Figure A-5 and A-6.

The values of the overrelaxation parameter, ORP, and the solution tolerence, DTMXA, are supplied on parameter card 3.

RESTRICTIONS:

MPLCT code in column 10 of parameter card 1 must be 1. IMPLSS must be called from $CENTRAL. The C array is not available to the user in the $PRETEMP operations block.

CALLING SEQUENCE: IMPLSS

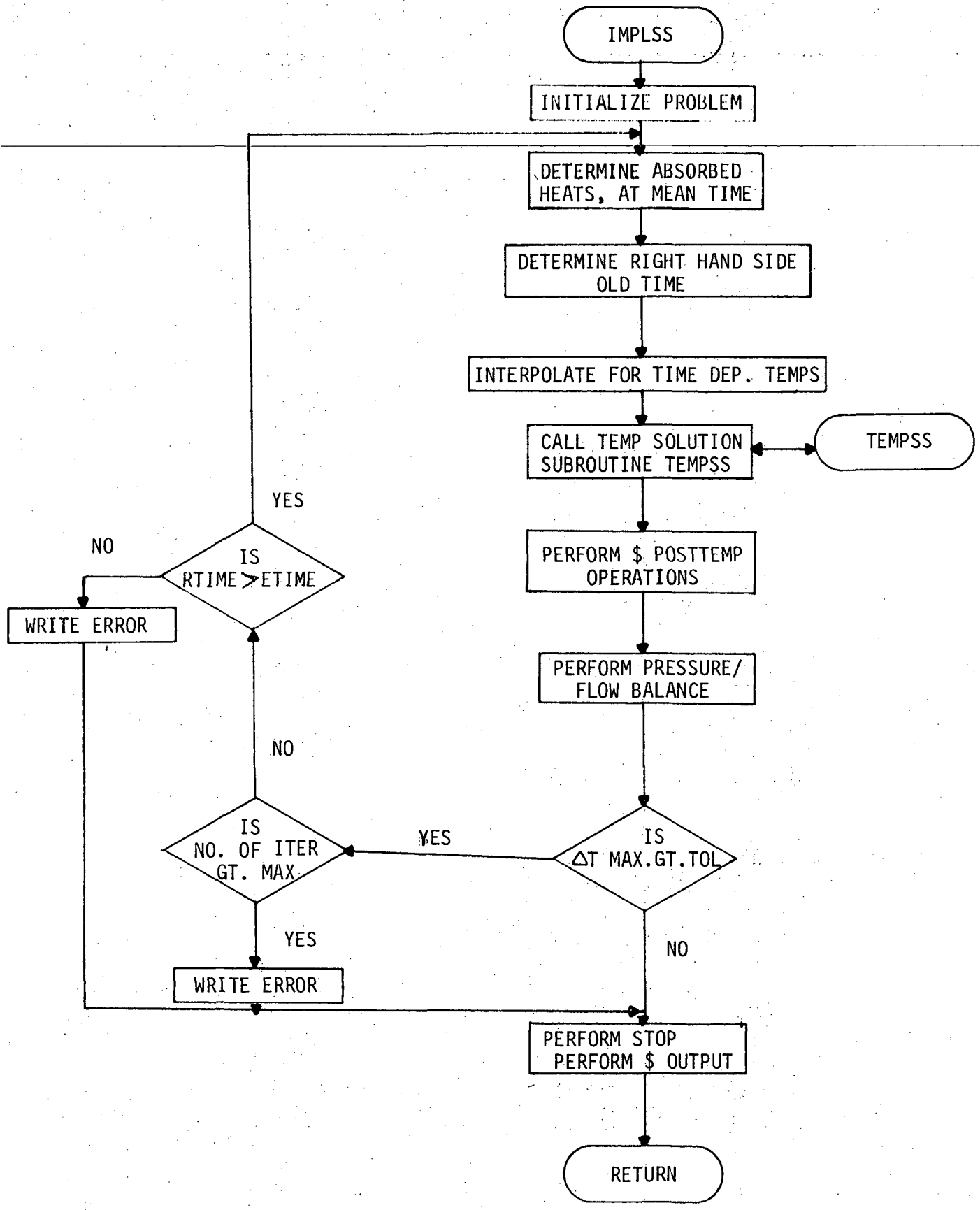


FIGURE A-5: FUNCTIONAL FLOW CHART OF IMPSS

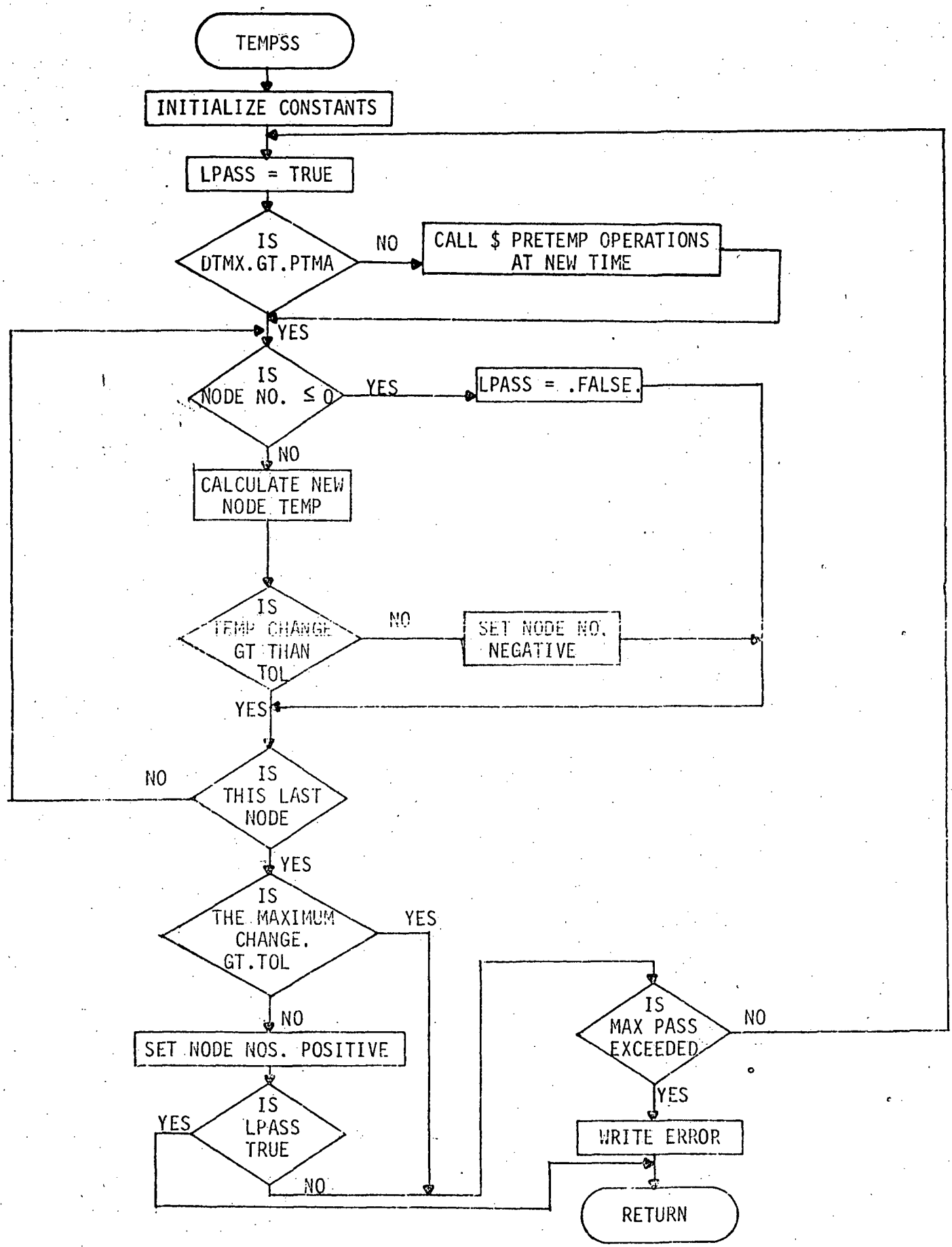


FIGURE A-6: FUNCTIONAL FLOW CHART OF TEMPSS

2.2 APPLICATION SUBROUTINES

SUBROUTINE NAME: RADIR

PURPOSE:

RADIR calculates the script-F values for infrared radiation heat transfer within an enclosure and uses these values to obtain the heat transfer during the problem. Several temperature nodes may be combined on a single surface for radiation heat transfer purposes. Also, the user may analyze problems with specular, diffuse or combinations of specular and diffuse radiation. See Section 3.1.3.5 for definitions and detailed description of methods.

RADIR calculates the script-F values on the initial call. This is performed by the procedure outlined in Section 3.1.3.5, Equations 23, 25 and 31. These values are stored for future use in the A6 array supplied by the user. The heat flux values are then calculated on all iterations by:

(1) Calculating the temperature of each surface using equation 34

(2) Calculating the absorbed heat for each node by the relation of equation 33

The value given by equation 35 is added to the conductor sum for each node so that the proper conveyence time increment may be obtained. As many enclosures as desired may be analyzed but each enclosure requires a different call to RADIR. RADIR must be called in $PRETEMP.

RESTRICTIONS:

Must be called from $PRETEMP

CALLING SEQUENCE:

```
RADIR (A1(IC),A2(IC),A3(IC),A4(IC),A5(IC),A6(IC))
```

Where the arrays are formated:

```
A1(IC),n,SN1,SA1,NN1,SN2,SA2,NN2,.....SNn,SAn,NNn
A2(IC),SE1,SE2-----SEn
A3(IC),SR1,SR2-----SRn
A4(IC),SNF1,SNT1,EFT1,SNF2,SNT2,EFT2,---SNFm,SNTm,EFTm
A5(IC),NNO(1,1),AN(1,1),NNO(1,2),AN(1,2)----NNO(1,NN1),AN(1,NN1),
       NNO(2,1),AN(2,1),NNO(2,2),AN(2,2)----NNO(2,NN2),AN(2,NN2),
         '        '       '        '          '          '
         '        '       '        '          '          '
         '        '       '        '          '          '
       NNO(n,1),AN(n,1),NNO(n,2),AN(n,2)----NNO(n,NNn),AN(n,NNn)
A6(IC), S, NSPACE
```

The following definitions apply in the above calling sequence.

A1,A2,....A6	Location for arrays supplied in the $CURVE block. The user must use the LUTAB function to find the location (see Section 5.2.5)
n	The number of surfaces
SN1,SN2,...SNn	Node number for surfaces - must be boundary nodes
SA1,SA2,....SAn	Total area for each surface
NN1,NN2,....NNn	Number of temperature nodes on each surface
SE1,SE2,....SEn	Emissivity values for each surface
SR1,SR2,....SRn	Diffuse reflectivity values for each surface
SNF1,SNT1,EFT1	Connections data: Surface number from, surface number to, E value from SNF1 to SNT1, etc.
NNO(X,Y)	Temperature node numbers on surfaces; Node number Y on surface X
NSPACE	Number of spaces needed to store script-FA values - NSPACE must be an integer values of n/2(n+1)
m	The number of surface connections

THERMAL RADIATION EXCHANGE

SUBROUTINE NAMES: IRRADI or IRRADE

PURPOSE:

These subroutines simulate a radiosity network* within a multiple gray diffuse surface enclosure containing a non-absorbing media. The input is identical for both subroutines. However, IRRADE utilizes explicit equations to obtain the solution by relaxation and IRRADI initially performs a symmetric matrix algebra inverse and thereafter obtains the exact solution implicitly by matrix multiplication. The relaxation criteria of IRRADE is internally calculated and severe enough so that both routines generally yield identical results. However, IRRADE should be used when temperature varying emissivities are to be considered and IRRADI should be used when the surface emissivities are constant. Both subroutines solve for the J node radiosity, obtain the net radiant heat flow rates to each surface and return them sequentially in the last array that was initially used to input the surface temperatures. The user need not specify any radiation conductors within the enclosure.

RESTRICTIONS:

The Fahrenheit system is required. The arbitrary number of temperature arguments may be constructed by a preceding BLDARY call. The emissivity, area, temperature-Q and upper half *FA* arrays must be in corresponding order and of exact length. The first data value of the *FA* array must be the integer number of surfaces and the second the Stefan-Boltzmann constant in the proper units and then the *FA* floating point values in row order. The diagonal elements (even if zero) must be included. As many radiosity subroutine calls as desired may be used. However, each call must have unique array arguments. The user should follow the radiosity routine by SCALE, BRKARY or BKARAD to distribute the Q's to the proper source location.

CALLING SEQUENCE: IRRADI(*AA(IC),Aε(IC),AFA(IC),ATQ(IC)*)
or IRRADE(*AA(IC),Aε(IC),AFA(IC),ATQ(IC)*)

where the arrays are formatted as follows:

AA(IC),A1,A2,A3,A4,..,AN
Aε(IC),ε1,ε2,ε3,ε4,...,εN
AFA(IC),N,σ,FA(1,1),FA(1,2),FA(1,3),FA(1,4),FA(1,5),..,FA(1,N)
FA(2,2),FA(2,3),FA(2,4),FA(2,5),..,FA(2,N)
. . . . .
FA(N-2,N-2),FA(N-2,N-1),FA(N-2,N)
FA(N-1,N-1),FA(N-1,N)
FA(N,N)

ATQ(IC),T1,T2,T3,..,TN

where *FA(1,2)* is defined as *A(1)*F(1,2)*. After the subroutine is performed the *ATQ* array is *ATQ(IC),Q1,Q2,Q3,..,QN*
Since $FA_1(1,2) \equiv FA_2(2,1)$ only the upper half triangle of the full *FA* matrix is required. IRRADI inverts this half matrix in its own area, hence approximately 300 surfaces may be considered using MOTAR on a 65K core machine.

*"Radiation Analysis by the Network Method," A. K. Oppenheim, Transaction of the ASME, May 1956, pp. 725-735.

SUBROUTINE NAME: RADSOL

PURPOSE:

RADSOL calculates a pseudo script-F for radiation from an external source entering an enclosure and uses these values to calculate the net heat transfer to each node due to the entering source. A number of temperature nodes may be combined on a single surface for radiation purposes. Also, problems with specular, diffuse, or combinations of specular and diffuse radiation may be analyzed. Section 3.1.3.5 should be consulted for definitions and descriptions of methods.

RADSOL calculates the pseudo script-F values on the initial call. This is performed by equations 38, 40, and 44 of section 3.1.3.5. The values are stored in the A7 array supplied by the user. The heat flux values are then calculated on each iteration by equations 45 and 46.

The user may analyze as many enclosures as desired by supplying a call statement for each enclosure. Also, a user may analyze several wave length bands of radiation for any enclosure by supplying a call statement (and appropriate data) for each wave length bands.

RESTRICTIONS:

Must be called from the $PRETEMP operations.

CALLING SEQUENCE:

```
RADSOL (A1(IC),A2(IC),A3(IC),A4(IC),A5(IC),A6(IC),A7(IC)
```

Where the arrays are formated:

```
A1(IC), n,SN1,SA1,NN1,SN2,SA2,NN2, - - - - - - Snn,SAn,NNn
A2(IC),SE1,SE2, - - - - - -SEn
A3(IC),SR1,SR2, - - - - - -SRn
A4(IC),SHT1,SHT2  - - - - -SHTn
A5(IC),SNF1,SNT1,EFT1,SNF2,SNT2,EFT2, - - -SNFm,SNTm,EFTm
A6(IC),NNO(1,1),AN(1,1),NNO(1,2),AN(1,2) - - -NNO(1,NN1),AN(1,NN1),
       NNO(2,1),AN(2.1),NNO(2,2),AN(2,2) - - -NNO(2,NN2),AN(2,NN2),
         '        '       '        '            '          '
         '        '       '        '            '          '
         '        '       '        '            '          '
       NNO(n,1),AN(n,1),NNO(n,2),AN(n,2) - - -NNO(n,NNn),AN(n,NNn)
A7(IC), S, NSPACE
```

The following definitions apply in the above calling sequence

A1,A2,.....A6	Location for arrays supplied in the $CURVE block. The user must use the LUTAE function to find the location (See Section 5.2.5)

n = the number of surfaces

SN1,SN2,...SNn	Node number for surfaces must be boundary nodes
SA1,SA2,...SAn	Total area for each surface
NN1,NN2,...NNn	Number of temperature nodes on each surface
SE1,SE2,...SEn	Emissivity values for each surface
SR1,SR2,...SRn	Diffuse reflectivity values for each surface
SHT1,SHT2,...SHTn	Incident heat flow on surfaces may be curve 1, 2, --- n or constant
SNF1,SNT1,EFT1	Connections data: Surface number from surface number to, E value from SNF1 to SNT1, etc.
NNO(X,Y)	Temperature node numbers on surfaces: Node number Y on surface X
NSPACE	Number of spaces needed to store script-FA values - NSPACE must be an integer values of n/2(n+1)

THERMAL RADIATION EXCHANGE

SUBROUTINE NAMES: SLRADI or SLRADE

PURPOSE:

These subroutines are very similar to IRRADI and IRRADE but are designed to solve for the solar heating rates within a enclosure. SLRADI inverts a half symmetric matrix in order to obtain implicit solutions, while SLRADE obtains solutions explicity by relaxation. SLRADE should be used when temperature varying solar absorptivities are to be considered. The second data value of the *AFA* array must be the solar constant in the proper units. The *AT* array allows the user to input the angle (degrees) between the surface normal and the surface-sun line. The *AI* array allows the user to input an illumination factor for each surface which is the ratio from zero to one of the unshaded portion of the surface. The solar constant (*S*), *AT* and *AI* values may vary during the transient for both routines. No input surface temperatures are required. The absorbed heating rates are returned sequentially in the *AQ* array, the user may utilize SCALE, BRKARY or BKARAD to distribute the heating rates to the proper source locations.

RESTRICTIONS:

These routines are independent of the temperature system being used. All of the array arguments must reference the integer count set by the SINDA preprocessor and be of the exact required length. As many calls as desired may be made but each call must have unique array arguments.

CALLING SEQUENCE: SLRADI(*AA(IC),Aε(IC),AFA(IC),AT(IC),AI(IC),AQ(IC)*)
or SLRADE(*AA(IC),Aε(IC),AFA(IC),AT(IC),AI(IC),AQ(IC)*)

SUBROUTINE NAME: EFFEMS

PURPOSE:

This subroutine calculates the effective emissivity *E* between parallel flat plates by the following equation:

$$E = 1.0/(1.0/E1 + 1.0/E1 - 1.0)$$

where *E1* and *E2* are the emissivities of the two surfaces under consideration.

RESTRICTIONS:

Arguments must be floating point numbers.

CALLING SEQUENCE: EFFEMS(*E1,E2,E*)

SUBROUTINE NAME: SCRPFA

PURPOSE:

To obtain the script FA value for radiant transfer within an enclosure. The input arrays are formatted as shown for subroutines IRRADI and IRRADE. The second data value in the *AFA* array is used as a final multiplier, if *1.0* the script *FA* values are returned; if σ then script σ *FA* values are returned. The script *FA* values are returned in the *ASFA* array which is formatted identical to the *AFA* array and may overlay it.

RESTRICTIONS:

All array arguments must reference the integer count set by the MOTAR preprocessor and all arrays must be exactly the required length.

CALLING SEQUENCE: SCRPFA(*AA(IC)*,*Aε(IC)*,*AFA(IC)*,*ASFA(IC)*)

NOTE: Subroutine SYMLST(*ASFA(IC)*+3,*ASFA(IC)*+1) may be called to list the matrix values and identify them by row and column number. This routine and the implicit radiosity routine finalize the half symmetric coefficient matrix and call on SYMINV(AFA(IC)+3,AFA(IC)+1) to obtain the symmetric inverse.

SUBROUTINE NAME: HXEFF

PURPOSE:

This subroutine obtains the heat exchanger effectiveness either from a user constant or from a biariant curve of effectiveness versus the flow rates on the two sides. The effectivenss thus obtained is used with the supplied flow rates, inlet temperatures and fluid properties to calculate the outlet temperatures using the methods described in Section 3.1.3.2. The user may specify a constant effectiveness by supplying a real number and may use the LUTAB*function to specify the effectiveness as a bivariant function of the two flow rates. The user also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for each of the two sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The LUTAB function may be used to obtain the specific heat values from a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:

HXEFF should be called in the $PRETEMP operations block. The value for EFF, the first argument must never be zero.

CALLING SEQUENCE:

HXEFF(EFF,W1,W2,CP1,CP2,TIN1,TIN2,TOOT1,TOOT2)

Where EFF - is (1) the effectiveness if real, (2) LUTAB (IEFF) where IEFF is a curve number of a bivariant curve of effectiveness versus W1 and W2. (See page A-60A)

W1,W2 - are the flow rates for side 1 and 2 respectively. May reference the flow rate array, W(I) where I is the tube number

CP1,CP2 - are the specific heat value for side 1 and side 2 fluid respectively. Constant values may be input or LUTAB may be used to reference curves in $CURVES

TIN1,TIN2 - are inlet lump temperatures - Usually T(LI1) and T(LI2) where LI1 and LI2 are the inlet lumps on side 1 and side 2

TOUT1,TOUT2 - are the outlet lump temperature locations where the calculated values will be stored

* See page 114

SUBROUTINE NAME: HXCNT

PURPOSE:

This subroutine calculates the heat exchanger using the relation described in Section 3.1.3.2 for a counter flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariant function of the two flow rates by using the LUTAB*function. The user also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for each of the two sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The LUTAB function may be used to obtain the specific heat values from a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:

HXCNT should be called in the $PRETEMP operations block. The value for UA, the first argument must never be zero.

CALLING SEQUENCE:

HXCNT(UA,W1,W2,CP1,CP2,TIN1,TIN2,TOUT1,TOUT2)

Where	UA	-	is (1) the heat exchanger conductance if real, (2) LUTAB (IUA) where IUA is a curve number of a bivariant curve of conductance versus W1 and W2 (See page A-60A)
	W1,W2	-	are the flow rates for sideland side 2 respectively. May reference the flow rate array, W(I) where I is the tube number
	CP1,CP2	-	are the specific heat values for side 1 and 2 fluid respectively. Constant values may be input or LUTAB may be used to reference curves in $CURVES
	TOUT1-TOUT2	-	are the outlet lump temperature locations where the calculated values will be stored

* See page 114

SUBROUTINE NAME: HXCROS

PURPOSE:

This subroutine calculates the heat exchanger using the relations described in Section 3.1.3.2 for a counter flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariant function of the two flow rates by using the LUTAB*function. Any one of the following four types of cross flow exchangers may be analyzed (see Section 3.1.3.2 for the relations):

1) Both steams unmixed
2) Both streams mixed
3) Stream with smallest MCp product unmixed
4) Stream with largest MCp product unmixed

The type is specified by the last argument in the call statement. Also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for both sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The LUTAB function may be used to obtain the specific heat values from a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:

HXCROS should be called in the $PRETEMP operations block. The value for UA, the first argument must never be zero.

CALLING SEQUENCE:

HXCROS(UA,W1,W2,CP1,CP2,TIN1,TIN2,TOUT1,TOUT2,K)

Where	UA	-	is (1) the heat exchanger conductance if real, (2) LUTAB (IUA) where IUA is a curve number of a bivariant curve of conductance versus W1 and W2.
	W1,W2	-	are the flow rates for side 1 and side 2 respectively. May reference the flow rate array, W(I) where I is the tube number
	CP1,CP2	-	are the specific heat values for side 1 and side 2 fluid respectively. Constant values may be input or LUTAB may be used to reference curves in $CURVES

* See page 114

TIN1,TIN2 - are inlet lump temperatures - Usually T(LI1) and T(LI2) where LI1 and LI2 are the inlet lumps on side 1 and side 2

TOUT1,TOUT2 - are the outlet lump temperature locations where the calculated values will be stored

K is the code specifying type of cross flow exchanger:

Both steams unmixed : K=1
Both streams mixed : K=2
Stream with small WCp Unmixed : K=3
Stream with large WCp unmixed : K=4

SUBROUTINE NAME: HXPAR

PURPOSE:

This subroutine calculates the heat exchanger using the relation described in Section 3.1.3.2 for a parallel flow type exchanger. The value of UA used in the calculations may be specified as a constant by supplying a real number or it may be specified as a bivariant function of the two flow rates by using the LUTAB*function. The user also supplies flow rates, specific heat values, inlet temperatures and a location for the outlet temperatures for each of the two sides. The flow rate array may be referenced to obtain flow rates and the temperature array may be used for temperatures. The LUTAB function may be used to obtain the specific heat values from a temperature dependent curve or a constant value may be supplied.

RESTRICTIONS:

HXPAR should be called in the $PRETEMP operations block. The value for UA, the first argument must never be zero.

CALLING SEQUENCE:

HXPAR(UA,W1,W2,CP1,CP2,TIN1,TIN2,TOUT1,TOUT2)

Where	UA	-	is (1) the heat exchanger conductance if real, (2) LUTAB (IUA) where IUA is a curve number of a bivariant curve of conductance versus W1 and W2. (See page A-60A)
	W1,W2	-	are the flow rates for side 1 and 2 respectively. May reference the flow rate array, W(I) where I is the tube number
	CP1,CP2	-	are the specific heat values for side 1 and side 2 fluid respectively. Constant values may be input or LUTAB may be used to reference curves in $CURVES
	TIN1,TIN2	-	are inlet lump temperatures - Usually T(LI1) and T(LI2) where LI1 and LI2 are the inlet lumps on side 1 and side 2
	TOUT1,TOUT2	-	are the outlet lump temperature locations where the calculated values will be stored

* See page 114

SUBROUTINE NAME: HEATER

PURPOSE:

This subroutine simulates an electrical heater with a control system which turns the heater on when the sensor lump temperature falls below the "heater on" temperature TON, and turns the heater off when the sensor lump rises above the heater off temperature, TOFF. When the heater is on, the input Q value is added to the Q location specified by the user. When the heater is off, the no heat is added.

RESTRICTIONS:

HEATER must be called in the $PRETEMP operations block.

CALLING SEQUENCE:

HEATER (TSEN,TON,TOFF,HT,Q)

Where TSEN is the sensed temperature
TON is the heater on temperature
TOFF is the heater off temperature
HT is the heater heat rate
Q is the location for storing the heat

SUBROUTINE NAME: CABIN

PURPOSE:

This subroutine performs a thermal and mass balance on a cabin air system. The cabin air is assumed to be a two component gas mixture with one condensible component and one noncondensible component. The cabin air is assumed to be well mixed so that the temperature and specific humidity are constant throughout. The cabin may contain any number of entering streams each with different temperature and humidity conditions. The cabin air may transfer heat to its surroundings any number of nodes with the heat transfer coefficient obtained by one of the three options:

1. User input coefficient
2. Relations for flow over a flat plot
3. Relations for flow over a tube bundle

The relations describing the second and third options are given in Section 3.1.3.4. The mass transfer coefficient for determing the rate of condensation or evaporation is determined by the Lewis relation which related the mass transfer coefficient directly to the connection heat transfer coefficient. By the Lewis Relation, if the diffusion coefficient is approximately wqual to the thermal diffusivity, the Sherwood number is approximately equal to the Nusselt number, thus giving a direct relation. (See Section 3.1.3.4 for details) Mass and heat transfer rates are determined at each node that interfaces the cabin gas as well at entering and exiting streams and a new cabin gas temperature and humidity is determined each iteration based upon the heat and mass balance. An account is kept of the condensate on the walls when condensation occurs but the condensate is assumed to remain stationary and not flow to other wall nodes.

Limits are applied when necessary to prevent more condensation than the vapor existing under severe transitent condition and to prevent evaporation of more liquid than exists at each wall lump.

As many cabins as desired may be analyzed in a given problem, but each must contain separate input information.

RESTRICTIONS:

CABIN must be called in $PRETEMP

CALLING SEQUENCE:

CABIN (A1, A2, A3, TC, A5, A6, A7, A8)

Where A1 is an array location * in $CURVES of an array containing the entering flow rate information. The format of the array is:

$NS, FR_1, PSI, TE_1, FR_2, PSI_2, TE_2$ ---- $FR_{ns}, PSI_{ns}, TE_{ns}$

A2 is an array location * in $CURVES of an array containing curve numbers in $CURVES for property values. The format of the array is:

NFLC,NMUO,NMUV,NCPO,NCPV,NKO,NKV,NLAT

A3 is an array location in $CURVES of an array containing pertinent constants. The format of the array is:

RA,RV,VC,PC,XC,WV,PSIC, PO, TO, CONV

T_c is the cabin gas temperature which must be a boundary node.

A5 is the location in $CURVES of an array containing node numbers and connection heat transfer coefficient values for nodes surrounding the cabin gas. The format of the array is:

LN_1, HA_1, LN_2, HA_2 - - - - - - LN_{n1}, HA_{n1}

A6 is the location in $CURVES of an array containing node numbers and information to permit calculation of convection coefficients for flat plates. The format is:

$LN_1, XX_1, XI_1, AI_1, VIW\emptyset_1, LN_2, XX_2, XI_2, AI_2,$

$VIN\emptyset_2$,-------$LN_{n2}, XX_{n2}, XI_{n2}, AI_{n2}, VIW\emptyset_{n2}$

A7 is the $CURVES location of an array containing node numbers and information to permit calculation of convection coefficients for tube bundles. The format is:

$LN_1, DI_1, AI_1, VIW\emptyset_1, LN_2, DI_2, AI_2, VIW\emptyset_2$,------$LN_{n3}$,

$DI_{n3}, AI_{n3}, VIWO_{n3}$

A8 is a working space array which must contain a number of spaces equivalent to three times the sum of the number of nodes with input heat transfer coefficients plus the number using flat plot relations plus the number using tube bundles.

* See page 114

The following symbol definitions apply in the above:

NS	Number of incoming streams
FR_i	Entering Flow rate for stream j
PSI_i	Specific humidity for entering stream i
TE_i	Temperature of entering stream i
NFLC	Curve number for circulation flow rate vs time
NMUO	Curve number for noncondensible viscosity vs temperature
NMUV	Curve number for condensible visocity vs temperature
NCPO	Curve number for noncondensible specific heat vs temperature
NCPV	Curve number for condensible specific heat vs temperature
NKO	Curve number for noncondensible thermal conduction vs temperature
NKV	Curve number for condensible thermal conduction vs temperature
NLAT	Curve number for latent heat of condensible vs temperature
RA	Gas constant for non-condensible component
RV	Gas constant for condensible component
VC	Cabin volume
PC	Cabin Pressure
XC	
WV	Initial vapor weight in cabin
PSIC	Initial specific humidity for cabin
LN_i	Cabin wall lump
HA	Heat transfer coefficient times area
n1	Number of wall lumps which have input HA values
n2	Number of wall lumps which have HA calculated by flat plate relations
n3	Number of wall lumps which have HA calculated by tube bundle relations
XX_i	Distance from leading edge for flat plate heating for ith flat plate node
XI_i	Length of flat plate in flow direction for ith flat plate node

AI_i	Heat transfer area for flat plate or tube node
DI_i	Tube outside diameter for tubes in the bundle for ith tube node
VIWO	Ratio of velocity at the lump to the circulation flow rate
To	The reference temperature to be used for estimating the saturation pressure of the condensible component. Should be near the range of saturation temperature expected
Po	The saturation pressure at To for the condensible component
CONV	Conversion factor to make the quantity XLAM/Rv/To dimensions less where XLAM is the latent heat of vaporization and Rv is the gas constant for the vapor. If XLAM is BTU/lb, Rv is FT-LB/°R and To is °R, CONV=778.

PHASE CHANGE

SUBROUTINE NAME: ABLATS

PURPOSE:

To provide a simple ablation (sublimation) capability for the SINDA user. The user constructs the 3-D network without considering the ablative. Then in $ POSTTEMP he simulates 1-D ablative attachments by calling ABLATS. ABLATS constructs the 1-D network and solves it by implicit forward-backward differencing (Crank-Nicholson method) using the time step set by the execution subroutine. Separate ablation arrays (AA) must be used for each ABLATS call. Required working space is obtained from unused program common. Several ABLATS calls thereby share unused common. The user must call subroutine PNTABL (AA) in the OUTPUT CALLS to obtain ablation totals and temperature distribution.

RESTRICTIONS:

ABLATS must be called in POSTTEMP and may be used with any execution subroutine. Subroutines DIDEG1 and NEWTR4 are called. All units must be consistent. The Fahrenheit system is required. Temperature varying material property arrays must not exceed 60 doublets. Bivariate material properties may be simulated by calling BVSPSA prior to ABLATS. Cross-sectional area is always considered unity. Thermal conductivity, Stefan-Boltzmann constant and density units must agree in area and length units.

CALLING SEQUENCE: ABLATS(*AA(IC),R,CP,G,T,C*)

where
- *C* is the capacitance location of the 3-D node attached to.
- *T* is the temperature location of the 3-D node attached to.
- *G* is the location of the material thermal conductivity or the starting location (integer count) of a doublet *G vs T* array.
- *CP* is the location of the material specific heat or the starting location (integer count) of a doublet C_p *vs T* array.
- *R* is the location of the material density or the starting location (integer count) of a doublet *R vs T* array.
- *AA(IC)* is the starting location of the ablation array which must be formatted as follows:

AA(IC)+1 *the ablative line number, a user specified identification integer.*

2 *integer number of sublayers (NSL) desired, ABLATS subtracts from this the number of sublayers ablated.*

3 *the initial temperature of the material, ABLATS replaces this with the outer surface temperature, always in degrees F.*

4 *the impressed outer surface heating rate per unit area, radiation rates not included.*

5 *material thickness; this is replaced by the sublayer thickness.*

6 *surface area of the 3-D node attached to, need not be unity.*

7 *ablation temperature, degrees F.*

8 *heat of ablation.*

9 *Stefan-Boltzmann constant in consistent units.*

10 *surface emissivity*

11 *space "sink" temperature, degrees F.*

12 SPACE,*N* *where N equals NSL + 4.*

NOTE: The outer surface radiation loss is integrated over the time step.

DYNAMIC STORAGE REQUIREMENTS:

This subroutine requires 3* (NSL+1) dynamic storage core locations.

PHASE CHANGE

SUBROUTINE NAME: LQSLTR

PURPOSE:

This subroutine accounts for the phase change energy of a melting or solidifying material. The temperature limits for the reaction must be specified (over at least a 1 degree range) and the phase change energy supplied as a constant rate over the range (Btu/°F). The network is constructed to include the capacitance effects of the phase change material. The network solution subroutines are allowed to calculate incorrect answers based on capacitance effects only; a call to LQSLTR in POSTTEMP then performs a corrector operation to account for any phase change occurring (reversability allowed) and returns corrected temperatures. The user is required to store the old temperature of the material (in POSTTEMP) and supply it as an argument to LQSLTR. This subroutine has a "DØ" loop built in and can be applied to several sequential nodes at once.

RESTRICTIONS:

The number of sequential nodes that this subroutine is to be applied to must be supplied as the integer *N*. All other arguments must be or address data values.

CALLING SEQUENCE: LQSLTR(*N,TL,TH,S(DV),C(DV),TØ(DV),TN(DV)*)

where
- *N* is the integer number of nodes to be operated on
- *TL* is the low temperature of the range
- *TH* is the high temperature of the range
- *S(DV)* is the first value of the phase change energy rate
- *C(DV)* is the first value of the nodal capacitances
- *TØ(DV)* is the first value of the old temperatures
- *TN(DV)* is the first value of the new temperatures

PHASE CHANGE

SUBROUTINE NAME: LQDVAP

PURPOSE:

This subroutine allows the user to simulate the addition of liquid to a node. The network data is prepared as though no liquid exists at the node and is solved that way by the network execution subroutine. Then LQDVAP, which must be called in POSTTEMP , corrects the nodal solution in order to account for the liquid. If the nodal temperature exceeds the boiling point of the liquid, it is set to the boiling point.

The excess energy above that required to reach the boiling point is calculated and considered as absorbed through vaporization. If the liquid is completely vaporized the subroutine deletes its operations. The method of solution holds very well for explicit solutions, but may introduce some error when large time steps are used with implicit solutions.

RESTRICTIONS:

This subroutine must be called in POSTTEMP .

CALLING SEQUENCE: LQDVAP (*T,C,A(IC)*)

where *T is the temperature location of the node.*
C is the capacitance location of the node.
A + 1 contains the initial liquid weight.
2 contains the liquid specific heat.
3 contains the liquid vaporization temperature.
4 contains the liquid heat of varporization.
5 receives the liquid vaporization rate (weight/time)
6 receives the liquid vaporization total (total weight)
7 contains the liquid initial temperature.

2.3 MATRIX SUBROUTINES

Input Format

Unless otherwise noted, the matrices require input as positive numbered arrays with integer number of rows and columns as the first two data values followed by floating point element values in row order.

Special Matrix Generation

Elemental Operations

SPECIAL MATRIX GENERATION

SUBROUTINE NAMES: ZERØ or ØNES

PURPOSE:

These subroutines generate a matrix [Z] such that every element is zero or one respectively.

RESTRICTIONS:

The matrix to be generated must contain exactly enough space in addition to having the integer number of rows and columns as the first two data values. The *NR* and *NC* arguments are the integer number of rows and columns respectively.

CALLING SEQUENCE: ZERØ(*NR,NC,Z(IC)*)

or ØNES(*NR,NC,Z(IC)*)

SUBROUTINE NAMES: UNITY or SIGMA

PURPOSE:

These are square matrix generation subroutines. UNITY generates a square matrix such that the main diagonal elements are one and all other elements are zero. SIGMA generates a square matrix such that all elements on and below the main diagnoal are one and the remaining elements are zero.

RESTRICTIONS:

The matrix [Z] to be generated must contain exactly enough space in addition to having the integer number of rows and columns as the first two data values. The integer number of rows and columns are equal and must be input as the argument *N*.

CALLING SEQUENCE: UNITY(*N,Z(IC)*)

or SIGMA(*N,Z(IC)*)

SUBROUTINE NAMES: GENALP or GENCØL

PURPOSE:

These are special matrix generation subroutines. GENALP will generate a matrix such that every element is equal to a constant *C*. GENCØL will generate a column matrix such that the first element is equal to *X1* and the last element is equal to *X2*. The intermediate elements receive equally incremented values such that a linear relationship is established between row number and element value.

RESTRICTIONS:

The *NR* and *NC* arguments refer to the integer number of rows and columns respectively. *X1*, *X2*,and *C* must be floating point values. The generated matrices must contain exactly enough space in addition to having the integer number of rows and columns as the first two data values.

CALLING SEQUENCE: GENALP(*NR,NC,C,Z(IC)*)

or GENCØL(*X1,X2,NR,Z(IC)*)

SPECIAL MATRIX FORMULATION

SUBROUTINE NAMES: FULSYM or SYMFUL

These subroutines allow the SINDA user to form a half symmetric matrix from a full square matrix or form a full square matrix from a half symmetric matrix, respectively. The arguments must address the matrix array integer count set by the preprocessor, the array lengths must be exact.

RESTRICTIONS:

The half symmetric matrix must be formatted as shown for subroutine IRRADI (Section 6.8) and the full square matrix must conform to the standard format.

CALLING SEQUENCE: FULSYM(*FM(IC)*,*SM(IC)*)

or SYMFUL(*SM(IC)*,*FM(IC)*)

Where *FM* is the full matrix and *SM* is the symmetric matrix.

SUBROUTINE NAME: SYMFRC

PURPOSE:

This subroutine may be used to force symmetry upon a square matrix. The main diagonal elements are untouched and all others are treated as follows:

$$x = (a_{ij} + a_{ji})/2.0;\ a_{ij} = x;\ a_{ji} = x$$

RESTRICTIONS:

The addressed matrix must be square and formatted as described in Section 4.2.2.3

CALLING SEQUENCE: SYMFRC(*A(IC)*)

SUBROUTINE NAMES: DIAG or UNDIAG or DIAGAD

PURPOSE:

Given a 1*N or N*1 matrix [*V*], subroutine DIAG forms a full square N*N matrix [*Z*]. The [*V*] values are placed sequentially on the main diagonal of [*Z*] and all off diagonal elements are set to zero. Subroutine UNDIAG forms a 1*N matrix [*V*] from the diagonal elements of an N*N matrix [*Z*]. Subroutine DIAGAD adds the elements of a 1*N matrix [*V*] to the diagonal elements of an N*N matrix [*Z*].

RESTRICTIONS:

Both matrices must have exactly enough space and contain their integer number of rows and columns as the first two data values.

CALLING SEQUENCE: DIAG(*V(IC)*,*Z(IC)*)

or UNDIAG(*Z(IC)*,*V(IC)*)

or DIAGAD(*V(IC)*,*Z(IC)*)

ELEMENTAL OPERATIONS

SUBROUTINE NAMES: ELEADD or ELESUB

PURPOSE:

These subroutines add or subtract the corresponding elements of two matrices respectively.

$$\underset{[Z]}{m*n} = \underset{[A]}{m*n} \pm \underset{[B]}{m*n}, \quad z_{ij} = a_{ij} \pm b_{ij}$$

RESTRICTIONS:

All matrices must be of identical size and have the integer number of rows and columns as the first two data values. The [Z] matrix may be overlayed into the [A] or [B] matrix.

CALLING SEQUENCE: ELEADD(*A(IC),B(IC),Z(IC)*)

or ELESUB(*A(IC),B(IC),Z(IC)*)

SUBROUTINE NAMES: ELEMUL or ELEDIV

PURPOSE:

These subroutines multiply or divide the corresponding elements of two matrices respectively.

$$\underset{[Z]}{m*n} = \underset{[A]}{m*n} \;*/\; \underset{[B]}{m*n}, \quad a_{ij} = a_{ij} \;*/\; b_{ij}$$

RESTRICTIONS:

All matrices must be of identical size and have the integer number or rows and columns as the first two data values. The [Z] matrix may be overlayed into the [A] or [B] matrix.

CALLING SEQUENCE: ELEMUL(*A(IC),B(IC),Z(IC)*)

or ELEDIV(*A(IC),B(IC),Z(IC)*)

SUBROUTINE NAME: ELEINV

PURPOSE:

This subroutine obtains the reciprocal of each element of the [A] matrix and places it in the corresponding element location of the [Z] matrix.

$$z_{ij} = 1.0/a_{ij}$$

RESTRICTIONS:

The matrices must be of identical size and have the integer number or rows and columns as the first two data values. The [Z] matrix may be overlayed into the [A] matrix.

CALLING SEQUENCE: ELEINV(*A(IC),Z(IC)*)

SUBROUTINE NAMES: EFSIN or EFASN

PURPOSE:

These subroutines perform elementary functions on all of the [A] matrix elements as follows:

$$z_{ij} = \sin(a_{ij}) \quad \text{or} \quad z_{ij} = \text{arcsine}(a_{ij})$$

RESTRICTIONS:

The matrices must be identical in size and have the integer number of rows and columns as the first two data values. The [Z] matrix may be overlayed into the [A] matrix.

CALLING SEQUENCE: EFSIN(*A(IC),Z(IC)*)

or EFASN(*A(IC),Z(IC)*)

SUBROUTINE NAMES: EFCØS or EFACS

PURPOSE:

These subroutines perform elementary functions on all of the [A] matrix elements as follows:

$$z_{ij} = \text{cosine}(a_{ij}) \quad \text{or} \quad a_{ij} = \text{arccosine}(a_{ij})$$

RESTRICTIONS:

The matrices must be identical in size and have the integer number of rows and columns as the first two data values. The [Z] matrix may be overlayed into the [A] matrix.

CALLING SEQUENCE: EFCØS(*A(IC),Z(IC)*)

or EFACS(*A(IC),Z(IC)*)

SUBROUTINE NAMES: EFTAN or EFATN

PURPOSE:

These subroutines perform elementary functions on all of the [A] matrix elements as follows:

$$z_{ij} = \text{tangent}(a_{ij}) \quad \text{or} \quad z_{ij} = \text{arctangent}(a_{ij})$$

RESTRICTIONS:

The matrices must be of identical size and have the integer number of rows and columns as the first two data values. The [Z] matrix may be overlayed into the [Z] matrix.

CALLING SEQUENCE: EFTAN(*A(IC),Z(IC)*)

or EFATN(*A(IC),Z(IC)*)

ELEMENTAL OPERATIONS

SUBROUTINE NAMES: EFABS or EFLØG or EFSQR

PURPOSE:

These subroutines perform elementary functions on all of the [*A*] matrix elements as follows respectively:

$$z_{ij} = |a_{ij}| \quad \text{or} \quad z_{ij} = \log_e(a_{ij}) \quad \text{or} \quad a_{ij} = \sqrt{a_{ij}}$$

RESTRICTIONS:

The matrices must be identical in size and have the integer number of rows and columns as the first two data values. All in the [*A*] matrix must be positive for EFLØG or EFSQR.

CALLING SEQUENCE: EFABS(*A(IC),Z(IC)*)

EFLØ (*A(IC),Z(IC)*)

EFSQR(*A(IC),Z(IC)*)

SUBROUTINE NAMES: EFEXP or EFPØW

PURPOSE:

These subroutines perform elementary functions on all of the [*A*] matrix elements as follows:

$$z_{ij} = e^{a_{ij}} \quad \text{or} \quad z_{ij} = a_{ij}^{\alpha}$$

RESTRICTIONS:

The matrices must be identical in size and have the integer number of rows and columns as the first two data values. The [*Z*] matrix may be overlayed into the [*A*] matrix. The exponent α may be an integer or floating point number. However, if any elements in [*A*] are negative then α must be an integer.

CALLING SEQUENCE: EFEXP(*A(IC),Z(IC)*)

or EFPØW(*A(IC),α,Z(IC)*)

ELEMENTAL OPERATIONS

SUBROUTINE NAMES: ADDALP or ALPHAA

PURPOSE:

To add a constant to or multiply a constant times every element in a matrix.

$$z_{ij} = C + a_{ij} \quad \text{or} \quad z_{ij} = C*a_{ij}$$

RESTRICTIONS:

The matrices must have exactly enough space and contain the integer number of rows and columns as the first two data values. C and all elements must be floating point numbers. The [Z] matrix may be overlayed into the [A] matrix.

CALLING SEQUENCE: ADDALP(*C,A(IC),Z(IC)*)

or ALPHAA(*C,A(IC),Z(IC)*)

SUBROUTINE NAMES: MATRIX or SCALAR or MATADD

PURPOSE:

The subroutine MATRIX allows a constant to replace a specific matrix element, subroutine SCALAR allows a specific matrix element to be placed into a constant location, and subroutine MATAAD adds a constant to a specific matrix element. The integers I and J designate the row and column position of the specific element.

$$z_{ij} = C \quad \text{or} \quad C = z_{ij} \quad \text{or} \quad z_{ij} = z_{ij} + C$$

RESTRICTIONS:

The matrix must have the integer number of rows and columns as the first two data values. Checks are made to insure that the identified element is within the matrix boundaries.

CALLING SEQUENCE: MATRIX(*C,I,J,Z(IC)*)

or SCALAR(*Z(IC),I,J,C*)

or MATADD(*C,I,J,Z(IC)*)

MATRIX OPERATIONS AND SOLUTIONS

SUBROUTINE NAME: **INVRSE**

PURPOSE:

To invert a square matrix.

$$\text{given } \underset{n*n}{[A]} \quad , \quad \underset{n*n}{[Z]} = \underset{n*n}{[A]^{-1}}$$

RESTRICTIONS:

The matrices must be square, identical in size and contain the integer number of rows and columns as the first two data values. The output matrix [*A*] may be overlayed into the [*A*] matrix.

CALLING SEQUENCE: INVRSE(*A(IC),Z(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

This subroutine requires n dynamic storage allocations.

SUBROUTINE NAME: **MULT**

PURPOSE: To multiply two conformable matrices together.

$$\underset{m*n}{[Z]} = \underset{m*p}{[A]} \quad \underset{p*n}{[B]} \quad , \quad z_{ij} = a_{ik} * b_{kj}$$

RESTRICTIONS:

The matrices must have exactly enough space and contain their integer number of rows and columns as the first two data values. If [*A*] and [*B*] are square, [Z] may be overlayed into either of them.

CALLING SEQUENCE: MULT(*A(IC),B(IC),Z(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

This subroutine requires n*m dynamic storage locations.

MATRIX OPERATIONS AND SOLUTIONS

SUBROUTINE NAME: TRANS

PURPOSE:

Given a matrix $\overset{m*n}{[A]}$ form its transpose as $\overset{n*m}{[Z]}$

RESTRICTIONS:

Both matrices must have exactly enough space and contain their integer number of rows and columns as the first two data values. The output matrix [Z] may be overlayed into the [A] matrix.

CALLING SEQUENCE: TRANS(*A(IC),Z(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

This subroutine requires n*m dynamic storage locations.

SUBROUTINE NAME: AABB

PURPOSE:

To sum two scaled matrices:

$$\overset{m*n}{[Z]} = C1\overset{m*n}{[A]} + C2\overset{m*n}{[B]}, \quad z_{ij} = C1*a_{ij} + C2*b_{ij}$$

RESTRICTIONS:

All matrices must be of identical size, contain exactly enough space and contain the integer number of rows and columns as the first two data values. The output matrix [Z] may be overlayed into either of the input matrices.

CALLING SEQUENCE: AABB(*C1,A(IC),C2,B(IC),Z(IC)*)

MATRIX OPERATIONS AND SOLUTIONS

SUBROUTINE NAMES: BTAB or BABT

PURPOSE:

To perform the following matrix operations, respectively:

$$\underset{n*m}{[Z]} = \underset{n*m}{[B]^t} \quad \underset{m*m}{[A]} \quad \underset{m*m}{[B]}$$

or

$$\underset{m*m}{[Z]} = \underset{m*n}{[B]} \quad \underset{n*n}{[A]} \quad \underset{n*m}{[B]^t}$$

RESTRICTIONS:

The matrices must be conformable, contain exactly enough space and contain the integer number of rows and columns as the first two data values. Subroutines MULT and TRANS are called on.

CALLING SEQUENCE: BTAB(*A(IC),B(IC),Z(IC)*)

or BABT(*A(IC),B(IC),Z(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

Due to subroutines MULT and TRANS this subroutine temporarily requires 2*m*n+6 dynamic locations.

SUBROUTINE NAMES: DISAS or ASSMBL

PURPOSE:

These subroutines allow a user to operate on matrices in a partitioned manner by disassembling a submatrix [*Z*] from a parent matrix [*A*] or assembling a submatrix [*Z*] into a parent matrix [*A*].

RESTRICTIONS:

The *I* and *J* arguments are integers which identify (by row and column number respectively) the upper left hand corner position of the submatrix within the parent matrix. All matrices must have exactly enough space and contain the integer number of rows and columns as the first two data values. The *NR* and *NC* arguments are the integer number of rows and columns respectively of the disassembled submatrix. If the submatrix exceeds the bounds of the parent matrix an appropriate error message is written and the program terminated.

CALLING SEQUENCE: DISAS(*A(IC),I,J,NR,NC,Z(IC)*)

or ASSMBL(*Z(IC),I,J,A(IC)*)

MATRIX OPERATIONS AND SOLUTIONS

SUBROUTINES NAMES: CØLMLT or RØWMLT

PURPOSE:

To multiply each element in a column or row of matrix [*A*] by its corresponding element from the matrix [*V*] which is conceptually a diagonal matrix but stored as a vector; i.e., 1*N or N*1 matrix. The matrix [Z] is the product.

RESTRICTIONS:

The matrices must have exactly enough space and contain the integer number of rows and columns as the first two data values. The matrices being multiplied must be conformable.

CALLING SEQUENCE: CØLMLT(*A(IC),V(IC),Z(IC)*)

or RØWMLT(*V(IC),A(IC),Z(IC)*)

SUBROUTINE NAMES: SHIFT or REFLCT

PURPOSE:

These subroutines may be used to move an entire matrix from one location to another. SHIFT moves the matrix exactly as is and REFLCT moves it and reverses the order of the elements within each column. The last element in each column becomes the first and the first becomes the last, etc.

RESTRICTIONS:

The matrices must be of identical size and the integer number of rows and columns must be the first two data values. The [Z] matrix may be overlayed into the [*A*] matrix.

CALLING SEQUENCE: SHIFT(*A(IC),Z(IC)*)

or REFLCT(*A(IC),Z(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

REFLCT uses three dynamic storage locations plus an additional one for each row.

SUBROUTINE NAME: SHUFL

PURPOSE:

This subroutine allows the user to reorder the size of a matrix as long as the total number of elements remains unchanged. The row order input matrix [*A*] is transposed to achieve column order and then reformed as a vector by sequencing the columns in ascending order. This vector is then reformed into a column order matrix by taking a column at a time sequentially from the vector. The newly formed column matrix is then transposed and output as the row order matrix [*Z*].

RESTRICTIONS:

The matrices must be identical in size and have their respective integer number of rows and columns as the first two data values. The number of rows times columns for [*A*] must equal the number of rows times columns of [*Z*].

CALLING SEQUENCE: SHUFL(*A(IC),Z(IC)*)

SUBROUTINE NAMES: CØLMAX or CØLMIN

PURPOSE:

These subroutines search an input matrix to obtain the maximum or minimum values within each column respectively. These values are output as a single row matrix [*A*] having as many columns as the input matrix [*A*].

RESTRICTIONS:

Each matrix must have its integer number of rows and columns as the first two data values.

CALLING SEQUENCE: CØLMAX(*A(IC),Z(IC)*)

or CØLMIN(*A(IC),Z(IC)*)

SUBROUTINE NAMES: SYMREM or SYMREP

PURPOSE:

These subroutines allow the SINDA user to operate on a single row/column of a half symmetric matrix. SYMREM will remove a particular row/column from the half symmetric matrix and place it into an array of the exact length to hold it. SYMREP will take an array and replace it into a specific row/column of the half symmetric matrix.

RESTRICTIONS:

The half symmetric matrix must be formatted as shown for subroutine IRRADI in Section 4.2.2.3. The integer *K* must designate the row/column to be operated on. If *K* is an integer zero, the main diagonal will be removed or replaced.

CALLING SEQUENCE: SYMREM(*K,SM(IC),A(IC)*)

or SYMREP(*K,A(IC),SM(IC)*)

SUBROUTINE NAME: SYMDAD

PURPOSE:

This subroutine will add the elements of a vector array to the corresponding elements of the main diagonal of a half symmetric matrix. If any of the elements is less than zero, they are set to zero.

RESTRICTIONS:

The half symetric matrix must be formatted as shown for subroutine IRRADI in Section 6.8. The vector array must be input as a positive array and be the same length as the matrix order.

CALLING SEQUENCE: SYMDAD(*VA(IC),SM(IC)*)

SUBROUTINE NAME: SYMINV

PURPOSE:

This subroutine obtains the inverse of a half symmetric matric matrix which is also symmetric and returns it in the same area as the input matrix. This subroutine is called internally by subroutines SCRPFA, IRRADI and SLRADI.

RESTRICTIONS:

This subroutine contains no error checks, exercise extreme caution when using it.

CALLING SEQUENCE: SYMINV(*A(DV),N*)

Where *A(DV)* addresses the 1,1 element and *N* is the matrix order.

SUBROUTINE NAME: PØLMLT

PURPOSE:

This subroutine performs the multiplication of a given number of n^{th} order polynomial coefficients by a similar number of m^{th} order polynomial coefficients. The polynomials must be input as matrices with the number of rows equal and each row receives the following operation:

$$(c_1,c_2,c_3,\ldots,c_k) = (a_1,a_2,\ldots,a_n) * (b_1,b_2,\ldots,b_m), k=m+n-1$$

RESTRICTIONS:

The matrices must have exactly enough space and contain their integer number of rows and columns as the first two data values.

CALLING SEQUENCE: PØLMLT(*A(IC),B(IC),C(IC)*)

SUBROUTINE NAME: PØLVAL

PURPOSE:

Given a set of polynomial coefficients as the first row of matrix [*A*], this subroutine evaluates the polynomial for the input complex number $X+iY$. The answer is returned as $U+iV$.

RESTRICTIONS:

[*A*] may be m*n but only the first row is evaluated.

CALLING SEQUENCE: PØLVAL(*A(IC),X,Y,U,V*)

SUBROUTINE NAME: PLYEVL

PURPOSE:

Given a matrix [*A*] containing an arbitrary number, NRA, of the n^{th} order polynomial coefficients and a column matrix [*X*] containing an arbitrary number, NRX, of *X* values, this subroutine evaluates each polynomial for each *X* value. The answers are output as a matrix [*Z*] of size NRX*NRA. Each set of polynomial coefficients in [*A*] is a row in ascending order. An *X* value evaluated for the polynomial creates a row in [*Z*] where the column number agrees with the polynomial row number.

RESTRICTIONS:

The matrices must have exactly enough space and contain their integer number of rows and columns as the first two data values.

CALLING SEQUENCE: PLYEVL(*A(IC),X(IC),Z(IC)*)

MATRIX OPERATIONS AND SOLUTIONS

SUBROUTINE NAME: **PØLSØV**

PURPOSE:

Given a set of polynomial coefficients as the first row in matrix [*A*], size (m,n+1), this subroutine calculates the complex roots which are returned as matrix [*Z*], size (n,2). Column 1 contains the real part and column 2 the imaginary part of the roots.

RESTRICTIONS:

This subroutine presently is limited to n = 20. It internally calls on RTPØLY and utilizes some double precision.

CALLING SEQUENCE: POLSOV(*A(IC),Z(IC)*)

SUBROUTINE NAME: **JACØBI**

PURPOSE:

This subroutine will find the eigenvalues [*E*] and eigenvector matrix [*Z*] associated with an input matrix [*A*].

n*n		n*n		n*n		n*n
[*A*]		[*Z*]	=	[*Z*]		[*E*]

RESTRICTIONS:

The matrices must have exactly enough space and contain their integer number of rows and columns as the first two data values. Note that matrix [*E*] is a diagonal matrix but is stated as a vector.

CALLING SEQUENCE: JACØBI(*A(IC),E(IC),Z(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

This subroutine requires 2*n*n+6 dynamic storage locations.

MATRIX DATA STORAGE AND RETRIEVAL

The ability to store and retrieve matrices from tape is easily achieved through the use of the FILE and CALL subroutines. Matrices are identified by an alphanumeric name, integer problem number and the core address of or for the matrix. The CALL subroutine searches the Matrix Input Tape and brings the desired matrix into core. The FILE subroutine writes a matrix onto the Matrix Output Tape. Subroutine ENDMØP causes all matrices from the Matrix Output Tape to be updated onto the Matrix Input Tape. In case of duplicate matrices, the one from the Output Tape replaces the one on Input Tape. A matrix which has been filed cannot be called until an ENDMØP operation has been performed. To create a new tape the user merely sets control constant NØCØPY nonzero and has a scratch tape mounted for the Input Tape. The user should check the section on control cards and deck setup to determine control card requirements.

SUBROUTINE NAMES: CALL or FILE

PURPOSE:

To allow the user to retrieve or store matrices on magnetic tape as described above. The *H* argument must be a six-character alphanumeric word and *N* must be an integer number, both of which are used to identify the matrix.

RESTRICTIONS:

See above. The matrix must have exactly enough space and contain the integer number of rows and columns as the first two data values.

CALLING SEQUENCE: CALL(*H*,*N*,*A*(*IC*))

or FILE(*A*(*IC*),*H*,*N*)

DYNAMIC STORAGE REQUIREMENTS:

Each of these routines requires 256 words of dynamic storage.

STORE AND RECALL

SUBROUTINE NAMES: ENDMØP or LSTAPE

PURPOSE:

Subroutine ENDMØP should be used in conjunction with subroutines CALL and FILE; see above. It causes matrices which have been filed by FILE on the Matrix Output Tape to be updated onto the Matrix Input Tape. A call to subroutine LSTAPE will cause the output of the name, problem number and size of every matrix stored on the Matrix Input Tape.

RESTRICTIONS:

See above.

CALLING SEQUENCE: ENDMØP

or LSTAPE

DYNAMIC STORAGE REQUIREMENTS:

Each of these routines requires 256 words of dynamic storage.

APPLICATION -- DYNAMIC VIBRATION

SUBROUTINE NAME: MØDES

PURPOSE:

This subroutine solves the following dynamic vibration equation

$$\underset{[A]}{m*n}\ \underset{[Z]}{m*n} = \underset{[B]}{n*n}\ \underset{[Z]}{n*n}\ \underset{\frac{1}{W^2}}{n*n}$$

where $[A]$ is the input inertia matrix associated with the kinetic energy and $[B]$ is the input stiffness matrix associated with the strain energy. $[Z]$ is the output eigenvector matrix associated with the frequencies of vibration W_i which are output in radians/sec as $[R]$ and in cycles/sec as $[C]$, both $[R]$ and $[C]$ are n*n diagonal matrices but stored as vectors.

RESTRICTIONS:

The matrices must have exactly enough space and contain their integer number of rows and columns as the first two data values. Subroutine JACØBI is called on.

CALLING SEQUENCE: MØDES(*A(IC),B(IC),Z(IC),R(IC),C(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

This subroutine requires 3*n*n+9 dynamic storage locations. An amount equal to 2*n*n+6 of these locations is required by subroutine JACØBI.

APPLICATION -- DYNAMIC VIBRATION

SUBROUTINE NAME: MASS

If a dynamic vibration problem is referred to a set of coordinates consisting of the deflections, ζ_i, and the rotations, θ_i, at N collocation points along the beam under consideration, then this subroutine generates the 2N by 2N inertia matrix [*A*] which appears in the following expression for kinetic energy:

$$T = \frac{1}{2} \left\{ \dot{\zeta}_1 \ldots \dot{\zeta}_n \dot{\theta}_1 \ldots \dot{\theta}_n \right\} [A] \begin{matrix} \dot{\zeta}_1 \\ \vdots \\ \dot{\zeta}_n \\ \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_n \end{matrix}$$

RESTRICTIONS:

The mass and inertia data input to this subroutine are to be supplied as piecewise continuous slices; however, these arrays may be of arbitrary size and different in length from each other. The number of collocation points, N, which determines the ultimate size, 2N by 2N, of the output inertia matrix, is also chosen arbitrarily.

CALLING SEQUENCE: MASS(*X(IC),DMPL(IC),RIPL(IC),CM(IC),A(IC)*)

where *X* is the matrix (N X 1) of collocation points referred to an arbitrary origin.

DMPL is the matrix (NDM X 4) of distributed mass per unit length slices, where
Col 1 is the location of the rear of a slice.
Col 2 is the location of the front of a slice.
Col 3 is the mass value at the rear of the slice.
Col 4 is the mass value at the front of the slice.

RIPL is the matrix (NRI X 4) of distributed rotary inertia per unit length slices. The columns here are similar to DMPL.

CM is the matrix (NCM X 4) of concentrated mass items, where
Col 1 is the attach point location for each item.
Col 2 is the mass at this location.
Col 3 is the location of its center of gravity.
Col 4 is the moment of inertia about the C. or G.

A is the output (2N X 2N) inertia matrix.

NOTE: Having application to *DMPL*, *RIPL* and *CM*, it is noted that the location of the values may not go beyond the limits of the collocation points in either direction.

APPLICATION -- DYNAMIC VIBRATION

SUBROUTINE NAME: **STIFF**

If a dynamic vibration problem is referred to a set of coordinates consisting of the deflections, ζ_i, and the rotations, θ_i, at N collocation points along the beam under consideration, then this subroutine generates the 2N by 2N stiffness matrix [*K*] which appears in the following expression for the strain energy:

$$U = \frac{1}{2} \left\{ \zeta_1 \cdots \zeta_n \theta_1 \cdots \theta_n \right\} [K] \begin{matrix} \zeta_1 \\ \vdots \\ \zeta_n \\ \theta_1 \\ \vdots \\ \theta_n \end{matrix}$$

RESTRICTIONS:

The stiffness and shear data input to this subroutine are to be supplied as piecewise continuous slices; however, these arrays may be of arbitrary size and different in length from each other. The number of collocation points, N, which determine the ultimate size, 2N by 2N, of the output stiffness matrix, is also chosen arbitrarily.

CALLING SEQUENCE: STIFF(*X(IC),EI(IC),GA(IC),K(IC)*)

where *X* is the matrix (N X 1) of collocation points referred to an arbitrary origin.

EI is the matrix (NEI X 4) of bending stiffness slices, where
Col 1 is the location of the rear of a slice.
Col 2 is the location of the front of a slice.
Col 3 is the stiffness value at the rear of a slice.
Col 4 is the stiffness value at the front of a slice.

GA is the matrix (NGA X 4) of shear stiffness slices, where the columns here are similar to those for the EI distribution.

K is the output stiffness matrix size 2N by 2N.

NOTE: Having application to *EI* and *GA*, it is noted that the location of the values may not go beyond the limits of the collocation points in either direction.

2.4 INTERPOLATION/EXTRAPOLATION SUBROUTINES

Lagrangian Interpolation

Linear Interpolation - Single Variable

Linear Interpolation - Two Single Variables

Parabolic Interpolation - Single Variable

The following are the formats for bivariant and trivarant arrays.

Bivariate

This type of array is used to represent a function of two independent variables: Z = (X,Y). Data values for a bivariate array are input in the following order:

```
n, X1, X2, . . ., Xn
Y1,Z11,Z12,. . ., Z1n
Y2,Z21,Z22,. . ., Z2n
.                  .
.                  .
.                  .
Ym,Zm1,Zm2,. . ., Zmn
```

where: n = Number of X values (integer)

m = Number of Y values (this value is not input explicitly)

Zji = f (Xi, Yj); X, Y, & Z = floating point values

Xi (i = 1,2,...,n) is strictly increasing in i.
Yj (j = 1,2,...,m) is strictly increasing in j.

The value of m is not input explicitly because the value of n (input as the first data value) and the number of points (generated by the preprocessor) are sufficient to define the location of any element in the array.

Trivariate Array

This type of array may be thought of as two or more bivariate arrays, where each bivariate array is associated with a third independent variable. Trivariate arrays are used to represent functions of the form F = f(X,Y,Z) for the purpose of evaluating such functions by interpolation. The data values in a trivariate array are input in the following order:

```
NX1,NY1,Z1,X1, X2,...,Xn
       Y1,F11,F12,..,F1n
       Y2,F21,F22,..,F2n        bivariate "sheet" for Z1
       .                .
       .                .
       Ym,Fm1,Fm2,..,Fmn
NX2,NY2,Z2,X1, X2,...,Xj
       Y1,F11,F12,..,F1j
       Y2,F21,F22,..,F2j
       .                .
       .                .
       Yk,Fk1,Fk2,..,FKj
NX3,NY3,Z3,...,...,...,...
```

A trivariate array may contain as many bivariate "sheets" as desired. The number of X and Y values in each sheet must be specified as integers NX and NY, respectively. NX and NY need not be the same for all sheets.

LAGRANGIAN INTERPOLATION

SUBROUTINE NAMES: LAGRAN or LGRNDA

PURPOSE:

These subroutines perform Lagrangian interpolation of up to order 50. The first requires one doublet array of X, Y pairs while the second requires two singlet arrays, one of X's and the other of Y's. They contain an extrapolation feature such that if the X value falls outside the range of the independent variable the nearest dependent Y variable value is returned and no error is noted.

$$Y = Pn\ (X) = \sum_{k=0}^{n} Yk \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{X - Xi}{Xk - Xi} \quad , n = 1,2,3,\ldots,50max.$$

RESTRICTIONS:

All values must be floating point except N which is the order of interpolation plus one and must be an integer. The independent variable values must be in ascending order.

CALLING SEQUENCE: LAGRAN(*X,Y,A(IC),N*)

or LGRNDA(*X,Y,AX(IC),AY(IC),Y*)

NOTE:

A doublet array is formed as follows:

X1,Y1,X2,Y2,X3,Y3,...,XN,YN

and singlet arrays are formed as follows:

X1,X2,X3,...,XN
Y1,Y2,Y3,...,YN

LINEAR INTERPOLATION - SINGLE VARIABLE

SUBROUTINE NAME: D1DEG1

PURPOSE:

This subroutine performs single variable linear interpolation on a doublet array of *X,Y* pairs.

RESTRICTIONS:

All values must be floating point numbers. The *X* independent variable values in the doublet array must be in ascending order.

CALLING SEQUENCE: D1DEG1(*X,A(IC),Y*)

where: *X = Input value of independent variable*
A = Doublet array of X,Y pairs
Y = Output value of dependent variable

SUBROUTINE NAME: D1D1DA

PURPOSE:

This subroutine performs single variable linear interpolation on a pair of singlet arrays containing corresponding values of *X* and *Y*.

RESTRICTIONS:

All values must be floating point numbers. The *X* independent variable values in the *AX* array must be in ascending order. The number of values in the *AX* and *AY* arrays must be the same.

CALLING SEQUENCE: D1D1DA(*X,AX(IC),AY(IC),Y*)

where: *X = Input value of the independent variable*
AX = Singlet array of X values
AY = Singlet array of Y values corresponding to the X values in AX
Y = Output value of the dependent variable

FUNCTION NAME: POL

PURPOSE:

This function subroutine performs single variable linear interpolation on a single array consisting of all the independent variables followed by all the dependent variables. The first location of the array contains the number of independent variables, the second contains an integer 1 (at the start) followed by all the independent variables and all the dependent variables. POL is useful for performing single variable interpolation on curves that are not type zero curves that are set up during the preprocessing phase.

RESTRICTIONS:

The first location must contain the number of independent variables and the second curve location contains the integer 1. The independent variables are next followed by the dependent variables. Note that POL is a function.

CALLING SEQUENCE: POL (A,X)

Where A is the array location

X is the value of the independent variable

Since POL is a function it must appear on the right of the equal of a Fortran statement.

LINEAR INTERPOLATION - SINGLE VARIABLE

SUBROUTINE NAMES: D1D1WM or D11MDA

PURPOSE:

These subroutines perform single variable linear interpolation by calling on D1DEG1 or D1D1DA respectively. However, the interpolated answer is multiplied by the values addressed as *Z* prior to being returned as *Y*.

RESTRICTIONS:

Same as D1DEG1 or D1D1DA and Z must be a floating point number.

CALLING SEQUENCE: D1D1WM(*X,A(IC),Z,Y*)

or D11MDA(*X,AX(IC),AY(IC),Z,Y*)

SUBROUTINE NAMES: D1MDG1 or D1M1DA

PURPOSE:

These subroutines use the arithmetic mean of two input values as the independent variable for linear interpolation. They require a doublet or two singlet arrays respectively.

RESTRICTIONS:

See D1DEG1 or D1D1DA as they are called on respectively.

CALLING SEQUENCE: D1MDG1(*X1,X2,A(IC),Y*)

or D1M1DA(*X1,X2,AX(IC),AY(IC),Y*)

LINEAR INTERPOLATION - SINGLE VARIABLE

SUBROUTINE NAMES: D1M1WM or D1M1MD

PURPOSE:

These subroutines use the arithmetic mean of two input values as the independent variable for linear interpolation. The interpolated answer is multiplied by the Z value prior to being returned as Y.

RESTRICTIONS:

Same as D1MDG1 or D1M1DA and Z must be a floating point number.

CALLING SEQUENCE: D1M1WM(*X1,X2,A(IC),Z,Y*) or D1M1MD(*X1,X2,AX(IC),AY(IC),Z,Y*)

SUBROUTINE NAMES: D1DG1I or D1D1IM or D1D1MI

PURPOSE:

These subroutines perform single variable linear interpolation on an array of *X*'s to obtain an array of *Y*'s. D1D1IM multiplies all interpolated values by a constant *Z* value while D1D1MI allows a unique *Z* value for each *X* value. They all call on D1DEG1.

RESTRICTIONS:

The number of input *X*'s must be supplied as the integer *N* and agree with the number of *Y* and *Z* locations where applicable. *Z* values must be floating point numbers.

CALLING SEQUENCE: D1DG1I(*N,X(DV),A(IC),Y(DV)*)

or D1D1IM(*N,X(DV),A(IC),Z,Y(DV)*)

or D1D1MI(*N,X(DV),A(IC),Z(DV),Y(DV)*)

SUBROUTINE NAMES: D11DAI or D11DIM or D11MDI

PURPOSE:

These subroutines are virtually identical to D1DG1I, D1D1IM and D1D1MI respectively. The difference is that they require singlet arrays for interpolation and call on D1D1DA.

RESTRICTIONS:

Same as D1DG1I, D1D1IM and D1D1MI.

CALLING SEQUENCE: D11DAI(*N,X(DV),AX(IC),AY(IC),Y(DV)*)

or D11DIM(*N,X(DV),AX(IC),AY(IC),Z,Y(DV)*)

or D11MDI(*N,X(DV),AX(IC),AY(IC),Z(DV),Y(DV)*)

LINEAR INTERPOLATION - SINGLE VARIABLE/TWO SINGLE VARIABLES

SUBROUTINE NAMES: D1IMD1 or D1IMWM or D1IMIM

PURPOSE:

These are indexed subroutines which use the arithmetic mean of two input values as the independent variable for linear interpolation. The array of answers (*Y*) produced are left as is (D1IMD1), are all multiplied by a single factor (D1IMWM), or each answer is multiplied by a separate factor.

RESTRICTIONS:

The interpolation array addressed must have an even number of input values and the independent variables must be in ascending order. These routines call up D1D1WM. *N* is the number of times the operation is to be performed.

CALLING SEQUENCE: D1IMD1(*N,X1(DV),X2(DV),A,Y(DV)*)

or D1IMWM(*N,X1(DV),X2(DV),A,Z,Y(DV)*)

or D1IMIM(*N,X1(DV),X2(DV),A,Z(DV),Y(DV)*)

LINEAR INTERPOLATION - TWO SINGLE VARIABLES

SUBROUTINE NAMES: CVQ1HT or CVQ1WM

PURPOSE:

These subroutines perform two single variable linear interpolations. The interpolation arrays must have the same independent variable *X* and dependent variables of, let's say, *R(X)* and *S(X)*. Additional arguments of *Y*, *Z* and *T* complete the data values. The post interpolation calculations are respectively:

$$Y = S(X)*(R(X)-T)$$

or $$Y = Z*S(X)*(R(X)-T)$$

RESTRICTIONS:

Interpolation arrays must be of the doublet type and have a common independent variable. All values must be floating point numbers.

CALLING SEQUENCE: CVQ1HT(*X,AR(IC),AS(IC),T,Y*)

or CVQ1WM(*X,AR(IC),AS(IC),T,Z,Y*)

PARABOLIC INTERPOLATION - SINGLE VARIABLE

SUBROUTINE NAMES: D1DEG2 or D1D2DA

PURPOSE:

These subroutines perform single variable parabolic interpolation. The first requires a double array of *X*, *Y* pairs while the second requires singlet arrays of *X* and *Y* values. They call on subroutines LAGRAN and LGRNDA respectively.

RESTRICTIONS:

See LAGRAN or LGRNDA respectively.

CALLING SEQUENCE: D1DEG2(*X,A(IC),Y*)

or D1D2DA(*X,AX(IC),AY(IC),Y*)

SUBROUTINE NAMES: D1D2WM or D12MDA

PURPOSE:

These subroutines perform single variable parabolic interpolation by calling on LAGRAN or LGRNDA respectively. However, the interpolated answer is multiplied by the value addressed as *Z* prior to being returned as *Y*.

RESTRICTIONS:

Same as LAGRAN or LGRNDA and *Z* must be a floating point number.

CALLING SEQUENCE: D1D2WM(*X,A(IC),Z,Y*)

or D12MDA(*X,AX(IC),AY(IC),Z,Y*)

PARABOLIC INTERPOLATION - SINGLE VARIABLE

SUBROUTINE NAMES: D1MDG2 or D1M2DA

PURPOSE:

These subroutines use the arithmetic mean of two input values as the independent variable for parabolic interpolation. They require a doublet or two singlet arrays respectively.

RESTRICTIONS:

See LAGRAN or LGRNDA as they are called on respectively.

CALLING SEQUENCE: D1MDG2(*X1,X2,A(IC),Y*)

or D1M2DA(*X1,X2,AX(IC),AY(IC),Y*)

SUBROUTINE NAMES: D1M2WM or D1M2MD

PURPOSE:

These subroutines use the arithmetic mean of two input values as the independent variable for parabolic interpolation. The interpolated answer is multiplied by the *Z* value prior to being returned as *Y*.

RESTRICTIONS:

Same as D1MDG2 or D1M2DA and *Z* must be a floating point number.

CALLING SEQUENCE: D1M2WM(*X1,X2,A(IC),Z,Y*)

or D1M2MD(*X1,X2,AX(IC),AY(IC),Z,Y*)

CYCLICAL INTERPOLATION ARRAYS

SUBROUTINE NAMES: D11CYL or DA11CY

PURPOSE:

These subroutines reduce core storage requirements for cyclical interpolation arrays. The arrays need cover one period only, and the period (*PR*) must be specified as the first argument. Linear interpolation is performed, and the independent variable must be in ascending order.

RESTRICTIONS:

All values must be floating point. Subroutine INTRFC is called on by both D11CYL and DA11CY, then D1DEG1 or D1D1DA respectively.

CALLING SEQUENCE: D11CYL(*PR,X,A(IC),Y*)

or DA11CY(*PR,X,AX(IC),AY(IC),Y*)

SUBROUTINE NAMES: D12CYL or DA12CY

PURPOSE:

These subroutines are virtually identical to D11CYL and DA11CY except that parabolic interpolation is performed.

RESTRICTIONS:

See D11CYL and DA11CY. Subroutines LAGRAN and LGRNDA respectively are called on.

CALLING SEQUENCE: D12CYL(*PR,X,A(IC),Y*)

or DA12CY(*PR,X,AX(IC),AY(IC),Y*)

CYCLICAL INTERPOLATION ARRAYS

SUBROUTINE NAMES: D11MCY or DA11MC

PURPOSE:

These subroutines are virtually identical to D11CYL and DA11CY except that the interpolation is multiplied by the floating point *Z* value prior to being returned as *Y*.

RESTRICTIONS:

Call on subroutines D1DEG1 and D1D1DA respectively.

CALLING SEQUENCE: D11MCY(*PR,X,A(IC),Z,Y*)

or DA11MC(*PR,X,AX(IC),AY(IC),Z,Y*)

SUBROUTINE NAMES: D12MCY or DA12MC

PURPOSE:

These subroutines are virtually identical to D11MCY and DA11MC except that parabolic interpolation is performed.

RESTRICTIONS:

Calls on subroutines LAGRAN and LGRNDA respectively.

CALLING SEQUENCE: D12MCY(*PR,X,A(IC),Z,Y*)

or DA12MC(*PR,X,AX(IC),AY(IC),Z,Y*)

POINT SLOPE INTERPOLATION

SUBROUTINE NAMES: GSLØPE

PURPOSE:

This subroutine will generate a slope array so that point slope interpolation subroutines can be used instead of standard linear interpolation subroutines. The user must address two singlet type arrays and a singlet slope array will be produced.

RESTRICTIONS:

The *X* independent variable array must be in ascending order. All arrays must be of equal length and contain floating point numbers.

CALLING SEQUENCE:

GSLØPE(*AX(IC),AY(IC),AS(IC)*)

SUBROUTINE NAMES: PSINTR or PSNTWM

PURPOSE:

These subroutines perform linear interpolation and require arrays of the *Y* points and slopes which correspond to the independent variable *X* array. All values must be floating point numbers. PSNTWM multiplies the interpolated answer by *Z* prior to returning it as *Y*.

RESTRICTIONS:

The independent *X* and dependent *Y* and slope arrays must be of equal length.

CALLING SEQUENCE:

PSINTR(*X,AX(IC),AY(IC),AS(IC),Y*)

or PSNTWM(*X,AX(IC),AY(IC),AS(IC),Z,Y*)

BIVARIATE INTERPOLATION

SUBROUTINE NAMES: BVSPSA or BVSPDA

PURPOSE:

These subroutines use an input *Y* argument to address a bivariate array* and pull off a singlet array of *Z*'s corresponding to the *X*'s or pull off a doublet array of *X*, *Z* values, respectively. The integer count for the constructed arrays must be exactly *N* or 2*N respectively. To use the singlet array for an interpolation call the *X* array can be reached by addressing the *N* in the bivariate array.

RESTRICTIONS:

As stated above, and all values must be floating point.

CALLING SEQUENCE: BVSPSA(*Y,BA(IC),AZ(IC)*)

or BVSPDA(*Y,BA(IC),AXZ(IC)*)

SUBROUTINE NAMES: BVTRN1 or BVTRN2

PURPOSE:

These subroutines construct a bivariate array of *Y*'s versus *X* and *Z* from an input bivariate array of *Z*'s versus *X* and *Y*. BVTRN1 should be used when the *Z* values increase with increasing *Y* values and BVTRN2 when the *Z* values decrease with increasing *Y* values.

RESTRICTIONS:

The user must appropriately place the *X* and *Z* values and spaces for *Y*'s in the array to be constructed. These subroutines will fill the *Y* spaces. The new array can differ in size from the old. Subroutine D1DEG1 is called and its linear extrapolation feature applies.

CALLING SEQUENCE: BVTRN1(*BAØ(IC),BAN(IC)*)

or BVTRN2(*BAØ(IC),BAN(IC)*)

* See page A-60A

BIVARIATE INTERPOLATION

SUBROUTINE NAMES: D2DEG1 or D2DEG2

PURPOSE:

These subroutines perform bivariate linear and parabolic interpolation respectively. The arrays must be formatted as shown for Bivariate Array Format.*

RESTRICTIONS:

For D2DEG1	,	$N \geq 2, M \geq 2$	See Bivariate
For D2DEG2	,	$N \geq 3, M \geq 3$	Array Format

CALLING SEQUENCE:

D2DEG1(*X,Y,BA(IC),Z*)

or D2DEG2(*X,Y,BA(IC),Z*)

SUBROUTINE NAMES: D2D1WM or D2D2WM

PURPOSE:

These subroutines perform bivariate* linear or parabolic interpolation by calling on D2DEG1 or D2DEG2 respectively. The interpolated answer is multiplied by the *W* value prior to being returned as *Z*.

RESTRICTIONS:

Same as D2DEG1 or D2DEG2 and *W* must be a floating point value.

CALLING SEQUENCE:

D2D1WM(*X,Y,BA(IC),W,Z*)

or D2D2WM(*X,Y,BA(IC),W,Z*)

* See page A-60A

BIVARIATE INTERPOLATION

SUBROUTINE NAMES: D2MXD1 or D2MXD2

PURPOSE:

These subroutines are virtually identical to D2DEG1 and D2DEG2 except that the arithmetic mean of two *X* values is used as the *X* independent variable for interpolation.

RESTRICTIONS:

Same as D2DEG1 or D2DEG2.

CALLING SEQUENCE: D2MXD1(*X1,X2,Y,BA(IC),Z*)

or D2MXD2(*X1,X2,Y,BA(IC),Z*)

SUBROUTINE NAMES: D2MX1M or D2MX2M

PURPOSE:

These subroutines are virtually identical to D2D1WM and D2D2WM except that the arithmetic mean of two *X* values is used as the *X* independent variable for interpolation.

RESTRICTIONS:

Same as D2D1WM and D2D2WM.

CALLING SEQUENCE: D2MX1M(*X1,X2,Y,BA(IC),W,Z*)

or D2MX2M(*X1,X2,Y,BA(IC),W,Z*)

TRIVARIATE INTERPOLATION

SUBROUTINE NAMES: D3DEG1 or D3D1WM

PURPOSE:

These subroutines perform trivariate linear interpolation. The interpolation array must be constructed as shown for Trivariate Array Format.* Subroutine D2DEG1 is called on which calls on D1DEG1. Hence, the linear extrapolation feature of these routines applies. Subroutine D3D1WM multiplies the interpolated answer by *F* prior to returning it as *T*.

RESTRICTIONS:

See Trivariate Array Format.* *F* must be a floating point value.

CALLING SEQUENCE: D3DEG1(*X,Y,Z,TA(IC),T*)

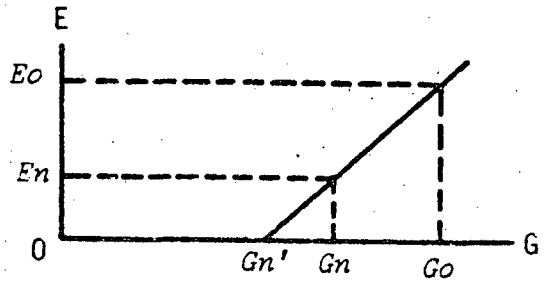
or D3D1WM(*X,Y,Z,TA(IC),F,T*)

LINEAR EXTRAPOLATION

SUBROUTINE NAME: ITRATE

PURPOSE:

Given two old guesses and their corresponding errors, this routine linearly extrapolates a new guess on the basis of zero error.



The new guess and error are positioned in the old locations and the extrapolated new guess is returned in the new guess location.

RESTRICTIONS:

If the error function being plotted has changes of slope, the user must insure that his guesses are quite accurate or divergence will be assured.

CALLING SEQUENCE: ITRATE(*EØ,GØ,EN,GN*)

* See page A-60A

2.5 OUTPUT SUBROUTINES

Network Printout

Floating Point

Array Printout

Plot Package

Tape Input/Output

Matrix Output

Special

Network Printout Subroutines

SUBROUTINE NAMES: TPRNT, CPRNT, QPRNT, UPRNT, DPRNT, or COPRNT

PURPOSE:

These subroutines provide the user with the ability to printout the elements of the thermal network. The purpose of each is listed below.

TPRNT	Prints all the thermal network temperatures in the T array
CPRNT	Prints all the thermal network capacitance in the C array
QPRNT	Prints all the values in the Q array
UPRNT	Prints all the conductance values in the U array
DPRNT	Prints all the values in the DTAU (time increment) array
COPRNT	Calls CPRNT, QPRNT, UPRNT and DPRNT

These subroutines are normally called in the OUTPUT block but may be called from other operations blocks. However, an excessive amount of output may occur if they are called from PRETMP or POSTMP.

RESTRICTIONS:

NONE

CALLING SEQUENCE: TPRNT, CPRNT, QPRNT, UPRNT, DPRNT or COPRNT

SUBROUTINE NAMES: WPRNT, PPRNT, or VPRNT

PURPOSE:

These subroutines provide the user with the ability to printout the fluid flow rate, the pressure of the pressure nodes, and valve positions. WPRNT prints flow rates; PPRNT prints pressures, and VPRNT prints valve positions. These subroutines are normally called from the OUTPUT block but may be called from other operations blocks if desired.

RESTRICTIONS: NONE

CALLING SEQUENCE: WPRNT, PPRNT, or VPRNT

FLOATING POINT PRINTOUT

SUBROUTINE NAMES: PRINT or PRINTL

PURPOSE:

These subroutines allow individual floating point numbers to be printed. The arguments may reference temperature, capacitance, source locations, conductors, or unique array locations. In addition, subroutine PRINTL allows each value to be preceded or labeled by a six-character alphanumeric word. The number of arguments is variable but the "label" array (*LA*) used for PRINTL should contain a Hollerith label for each argument.

RESTRICTIONS:

Integers must first be floated.

CALLING SEQUENCE: PRINT(*T,C,Q,G,K,...,A+*)

or PRINTL(*LA(DV),T,C,Q,G,K,...,A+*)

ARRAY PRINTOUT

SUBROUTINE NAME: PRINTA

PURPOSE:

This subroutine allows the user to print out an array of values, five to the line. The integer array length *N* and the first data value location must be specified. Each value receives an indexed label. The user must supply a six-character alphanumeric word *L* to be used as a common label and an integer value *M* to begin the index count.

RESTRICTIONS:

The array values to be printed must be floating point numbers. If *L* is supplied as a literal Hollerith data value (instead of a reference to a user constant containing same), it must be entered in FORTRAN-compatible H-type notation (e.g., 4HTEMP).

CALLING SEQUENCE: PRINTA(*L,A(DV),N,M*)

If the label was the word 'TEMP', *N* was 3 and *M* was 6, the line of output would look as follows:

TEMP (6)*value* TEMP (7)*value* TEMP (8)*value*

SUBROUTINE NAME: PRNTMA or PRNTMI

PURPOSE:

This subroutine allows the user to print out up to 10 arrays in a column format. The individual elements are not labeled but each column receives a two-line heading of 12 alphanumeric characters/line. The two-line heading must be supplied as a single array of four words, six characters each. The user must supply the starting location of each label array and value array. The number of values in each value array must agree and be supplied as the integer *N*. The value arrays must contain floating point numbers.*

RESTRICTIONS:

Labels must be alphanumeric while values must be floating point.* All floating point value arrays must contain the same number of values.

CALLING SEQUENCE: PRNTMA(*N,LA1(DV),VA1(DV),LA2(DV),VA2(DV),...*)

PRNTMI(*N,LA1(DV),VA1(DV),LA2(DV),VA2(DV),...*)

**VA1* only must address an array of integers for subroutine PRNTMI.

ARRAY PRINTOUT

SUBROUTINE NAME: PUNCHA

PURPOSE:

This subroutine enables a user to punch out an array of data values in any desired format. The *F* argument must reference a FØRTRAN FØRMAT which has been input as an array, including the outer parenthesis but deleting the word FØRMAT.* The second argument must address the first data value of the array of sequential values. The third argument, *N*, must be the integer number of data values in the array.

CALLING SEQUENCE: PUNCHA(*F(DV),A(DV),N*)

SUBROUTINE NAMES: GENOUT, GENI or GENR

PURPOSE:

These subroutines print out arrays of numbers 10 to a line. GENOUT prints either real numbers, integer or both. GENI and GENR print integers and real number arrays respectively. The integer are written in a I9 format and the real numbers in a E12.4 format.

RESTRICTIONS:

GENI writes arrays of integers only. GENR writes arrays of real numbers only.

CALLING SEQUENCE:

```
GENOUT  (A, ISTRT, ISTP,'NAME')
GENI    (A, ISTRT, ISTP,'NAME')
GENR    (A, ISTRT, ISTP, 'NAME')
```

Where A is the array location
ISTRT is the first value in A being written
ISTP is the last value in A being written
'NAME' is a title of 22 Hallerith words for identification

PLOT PACKAGE

SUBROUTINE NAME: PRNPLT

PURPOSE:

This subroutine will print out a plot of data on the line printer. It is intended primarily for plotting temperature histories which were accumulated in the OUTPUT CALLS block. One or two curves of up to 100 points each may be plotted on each frame (page). Y-axis scaling is automatic. No units are associated with the X-axis, and no X values are used; one point is plotted for each print wheel position along the X-axis. Points on the first curve will be printed as 'X's, and points on the second curve will be printed as 'O's. Where points overlap, an asterisk, '*', is printed.

RESTRICTIONS:

If *NA* and/or *NB* is greater than 100, only the first 100 points in the corresponding array of Y-values (*YA* and/or *YB*) will be plotted. The argument *LP* normally has a value of 50 when standard 11 x 14 computer paper is used. The smallest Y increment represented by a line is one unit, so the narrowest range covered by the Y axis will be *LP* units. One graph only may be plotted on a single page. When a point to be plotted has a value which lies exactly between the values associated with two adjacent lines, then the point will be printed on both lines.

CALLING SEQUENCE: PRNPLT(*TT(DV),NA,YA(DV),TA(DV),NB,YB(DV),TB(DV),LP*)

where:
- *TT = Main title (4 Hollerith words)*
- *NA = Number of points for the first curve (integer; must be greater than zero)*
- *YA = Y values for the first curve*
- *TA = Title for the first curve (4 Hollerith words)*
- *NB = Number of points for the second curve (may be zero)*
- *YB = Y values for the second curve*
- *TB = Title for the second curve (4 Hollerith words)*
- *LP = Number of printer lines which may be used to plot values (at least two less than the number of lines on a page)*

PLOT PACKAGE

SUBROUTINE NAMES: PLØTX1 or PLØTX2 or PLØTL1 or PLØTL2

PURPOSE:

These FØRTRAN V coded quick plot subroutines call upon a large package of undocumented subroutines specifically for the SC-4060. They will produce up to four graphs per frame and several variables may be plotted per graph. A suitable grid will be drawn with certain lines emphasized. The grid lines will have reasonable numerical indicia and centered title will be printed for both axes and at the top of the graph.

PLØTX1 and PLØTL1 will compute the minimum and maximum values of the stored X and Y arrays to be plotted and call upon PLØTX2 or PLØTL2 which use the values as grid limits for the graph. The user may set the grid limits by calling PLOTX2 and PLOTL2 directly. The X, Y and top titles (*XT*, *YT* and *TT* respectively) must consist of nine alphanumeric words of six characters each.

RESTRICTIONS:

The user should consult Appendix D to check tape designation requirements. The X and Y values must be floating point numbers. The user must call subroutine PLTND after all his plotting is done. No limit may be zero for log plots.

CALLING SEQUENCE:

PLØTX1(*N,IS,TX(DV),TY(DV),TT(DV),NP,AX(DV),AY(DV)*)

or

PLØTX2(*N,XL,XR,YB,YT,IX,TX(DV),TY(DV),TT(DV),NP,AX(DV),AY(DV)*)

or

PLØTL1(*N,IS,TX(DV),TY(DV),TT(DV),NP,AX(DV),AY(DV),LM*)

PLØTL2(*N,XL,XR,YB,YT,IS,TX(DV),TY(DV),TT(DV),NP,AX(DV),AY(DV),LM*)

where:

N	*is the integer number of graphs per frame (1,2,3 or 4); if zero, the grid from the previous plot call is used.*
IS	*is the integer identifying the plotting symbol (1-144)*
TX	*is the address of the X title*
TY	*is the address of the Y title*
TT	*is the address of the top title*
NP	*is the integer number of XY values or points to be plotted; if negative the points will be connected by straight lines.*
AX	*is the address of the X array*
AY	*is the address of the Y array*
XL	*is the floating point X axis left limit*
XR	*is the floating point X axis right limit*
YB	*is the floating point Y axis bottom limit*
YT	*is the floating point Y asis top limit*
LM	*is an integer identifying the log plotting mode; if less than zero plot log X versus linear Y, if equal to zero plot log X versus log Y, if greater than zero plot linear X versus log Y*

PLOT PACKAGE

SUBROUTINE NAMES: PLØTX3 or PLØTX4

PURPOSE:

These subroutines are similar to PLØTX1 and PLØTX2 but have six additional arguments which allow the user to modify the grid as desired.

RESTRICTIONS:

See PLØTX1 and PLØTX2.

CALLING SEQUENCE:

PLØTX3(*N,IS,TX(DV),TY(DV),TT(DV),NP,AS(DV),AY(DV),DX,DY,L,M,I,J*)

or

PLØTX4(*N,XL,XR,YB,YT,IS,TX(DV),TY(DV),TT(DV),NO,AX(DV),AY(DV),DX,DY,LM,I,J*)

where the arguments are identical to PLØTX1 and PLØTX2 except for

DX,DY these floating point values are used in spacing the grid lines which are centered on the zero values. If zero, no grid lines will be drawn.

L,M these integers cause every L^{th} vertical and M^{th} horizontal grid line to be redrawn for emphasis. If zero, no grid lines will be emphasized. If negative, a square grid will be produced.

I,J these integers cause every I^{th} vertical and J^{th} horizontal grid line to be labeled with its value. If zero, no grid lines will be labeled. If negative, the labels will be placed outside the grid, otherwise they will appear on the zero axis.

SC-4060 PLOT SYMBOL DICTIONARY

(for use with quick plot subroutines only)

Integer	Symbol	Integer	Symbol	Integer	Symbol	Integer	Symbol
1	A	31	4	61	m	105	∫
2	B	32	5	62	n	106	∂
3	C	33	6	63	o	107	○
4	D	34	7	64	p	108	<
5	E	35	8	65	q	109	#
6	F	36	9	66	r	110	(logical inverse)
7	G	37	(blank)	67	s	111	|
8	H	38	.	68	t	112	π
9	I	39	,	69	u	113	_
10	J	40	'(close quote)	70	v	114	□
11	K	41	$	71	w	115	Σ
12	L	42	(	72	x	116	(tilde)
13	M	43	)	73	y	117	(lozenge)
14	N	44	/	74	z	118	Δ
15	Ø	45	-(minus)	88	"	121	←
16	P	46	+	89	¢	122	→
17	Q	47	*	90	[	123	o(circle)
18	R	48	=	91	]	124	·
19	S	49	a	92	?	125	.
20	T	50	b	93	-(hyphen)	126	•
21	U	51	c	94	!	127	●
22	V	52	d	95	;	136	'(open quote)
23	W	53	e	96	:	138	{
24	X	54	f	97	α	139	}
25	Y	55	g	98	β	140	\
26	Z	56	h	99	^(caret)	141	-(bar)
27	0	57	i	100	δ	142	±
28	1	58	j	102	%	143	@
29	2	59	k	103	γ	144	&
30	3	60	l	104	>		

PLOT PACKAGE:

SC-4020 PLOT SYMBOL DICTIONARY

(to be used at installations, such as NASA/MSC, where an SC-4060 is used to simulate an SC-4020)

Decimal Integer	Plot Char.	Decimal Integer	Plot Char.	Decimal Integer	Plot Char.	Decimal Integer	Plot Char.
0	0	16	+	32	-	48	
1	1	17	A	33	J	49	/
2	2	18	B	34	K	50	S
3	3	19	C	35	L	51	T
4	4	20	D	36	M	52	U
5	5	21	E	37	N	53	V
6	6	22	F	38	Ø	54	W
7	7	23	G	39	P	55	X
8	8	24	H	40	Q	56	Y
9	9	25	I	41	R	57	Z
10	∂	26	π	42	.	58	°
11	=	27	.	43	$	59	,
12	"	28	)	44	*	60	(
13	'	29	β	45	γ	61	∫
14	δ	30	I	46	~	62	Σ
15	α	31	?	47	d	63	□

TAPE INPUT/OUTPUT

SUBROUTINE NAMES: READ or WRITE

PURPOSE:

These subroutines enable the user to read and write arrays of data as binary information on magnetic tape. The first argument *L* must be the integer number of the logical tape being addressed. The second argument *X* must address the first data value of the array to be written out or starting location for data to be read into. The third argument *N* must be an integer For WRITE, it is the number of data values to be written on tape as a record. For READ, it is the number of data values to be read in from tape from the next record, not necessarily the entire record.

RESTRICTIONS:

The user should check Appendix D to determine which logical tapes are available and control card requirements. All processed information must be in binary.

CALLING SEQUENCE: READ(*L,X(DV),N*)

or WRITE(*L,X(DV),N*)

MATRIX PRINTOUT

SUBROUTINE NAME: LIST

PURPOSE:

This subroutine prints the elements of a matrix [*A*] and identifies each by its row and column number. The user must supply an alphanumeric name *ALP* and integer number *NUM* to identify the matrix. This is to maintain consistency with subroutines FILE and CALL.

RESTRICTIONS:

The matrix must have its integer number of rows and columns as the first two data values.

CALLING SEQUENCE: LIST(*A(IC),ALP,NUM*)

SUBROUTINE NAME: PUNCH

PURPOSE:

This subroutine punches out a matrix.[*A*],.size n*m, one column at a time in any desired format. The argument FØR must reference a FØRTRAN format statement that has been input as a positive array. It must include the outer parenthesis but not the word FØRMAT. The argument *HEAD* must be a single BCD word used to identify the matrix. Each column is designated and restarts use of the FØRMAT statement.

RESTRICTIONS:

The matrix [*A*] must have exactly enough space and contain the integer number of rows and columns as the first two data values.

CALLING SEQUENCE: PUNCH(*A(IC),HEAD,FØR(IC)*)

DYNAMIC STORAGE REQUIREMENTS:

This subroutine required n+3 dynamic storage locations.

SUBROUTINE NAME: SYMLST

PURPOSE:

To print out and identify the element values of a half symmetric matrix. This output subroutine is most generally used with subroutine SCRPFA.

CALLING SEQUENCE: SYMLST(*A(DV),N*)

where *A(DV)* adresses the 1,1 element and *N* is the matrix order.

SPECIAL

SUBROUTINE: PNTABL

PURPOSE:

To provide output information for users of subroutine ABLATS. The ABLATS routine performs ablative simulation calculations but since it is called in $ POSTTEMP, it performs no output. The user must call PNTABL in the $ OUTPUT block and reference the ablative array of the ABLATS call. When the ablative material is expended, ABLATS will call PNTABL directly and will also cause current problem time to be printed.

RESTRICTIONS:

This routine is called in conjunction with subroutine ABLATS only, see Section 2.2

CALLING SEQUENCE: PNTABL(*AA(IC)*)

2.6 MATHEMATICAL SOLUTION SUBROUTINES

Area Integration

Polynomial/Simultaneous Linear Equations

Curve Fit/Temperature Derivative

Complex Variable Analysis

AREA INTEGRATION

SUBROUTINE NAMES: SMPINT or TRPZD

PURPOSE:

These subroutines perform area integrations by Simpson's rule and the trapezoidal rule respectively. Simpson's rule requires that an odd number of points be supplied. If an even number of points is supplied, SMPINT will apply the trapezoidal rule to the last incremental area but Simpson's rule elsewhere. The respective operations are:

$$A = DX*(Y1+4Y2+2Y3+4Y4+...+YN)/3$$

or $$A = DX*(Y1+2Y2+2Y3+2Y4+...+YN)/2$$

RESTRICTIONS:

The *DX* increment must be uniform between all the *Y* points. All values must be floating point except *N* which must be an integer.

CALLING SEQUENCE: SMPINT(*N*,*DX*,*Y(DV)*,*A*)

or TRPZD(*N*,*DX*,*Y(DV)*,*A*)

SUBROUTINE NAME: TRPZDA

PURPOSE:

This subroutine performs area integration by the trapezoidal rule. It should be used where the *DX* increment is not uniform between the *Y* values but the corresponding *X* value for each *Y* value is known. The operation performed is as follows:

$$A = \frac{1}{2} \Sigma\ (Xi-Xi-1)*(Yi+Yi-1) \quad , \quad i = 2,N$$

All values must be floating point numbers except the array length *N* which must be an integer.

CALLING SEQUENCE: TRPZDA(*N*,*X(DV)*,*Y(DV)*,*A*)

ROOTS

SUBROUTINE NAMES: NEWTRT or NEWRT4

PURPOSE:

These subroutines utilize Newton's method to obtain one root of a cubic or quartic equation respectively. The root must be in the neighborhood of the supplied initial guess and up to 100 iterations are performed in order to obtain an answer within the specified tolerance. If the tolerance is not met, an answer of 10^{38} is returned. The respective equations are:

$$f(X) = A1+A2*X+A3*X^2+A4*X^3 = 0.0 \pm T$$

or

$$g(X) = A1+A2*X+A3*X^2+A4*X^3+A5*X^4 + 0.0 \pm T$$

where X starts as the initial guess *RI* and finishes as the final answer *RF*. *T* is the tolerance.

RESTRICTIONS:

All data values must be floating point numbers.

CALLING SEQUENCE: NEWTRT(*A(DV),T,RI,RF*)

OR NEWRT4(*A(DV),T,RI,RF*)

POLYNOMIAL/SIMULTANEOUS LINEAR EQUATIONS

SUBROUTINE NAMES: PLYNML or PLYARY or PLYAWM

PURPOSE:

These subroutines calculate Y from the following polynomial equation:

$$Y = A1+A2*X+A3*X^2+A4*X^3+\ldots+AN+1*X^N$$

$$Z = Y*W$$

The number of terms is variable but all the A coefficients must be input no matter what their value.

RESTRICTIONS:

All values must be floating point numbers except for the degree of polynomial N which must be integer.

CALLING SEQUENCE: PLYNML($X,A1,A2,A3,\ldots,AN,Y$)

or PLYARY($N,X,A(DV),Y$)

or PLYAWM($N,X,A(DV),W,Z$)

SUBROUTINE NAME: SIMEQN

PURPOSE:

This subroutine solves a set of up to 10 linear simultaneous equations by the factorized inverse method. The problem size and all input and output values are communicated as a single specially formatted positive input array. The array argument must address the matrix order (N) which is input by the user. The first data value must be the integer order of the set (or size of the square matrix) followed by the coefficient matrix $[A]$ in column order, the boundary vector $\{B\}$ and space for the solution of vector $\{S\}$.

$$[A] \quad \{S\} = \{B\}$$

RESTRICTIONS:

The integer count and matrix size must be integers, all other values must be floating point. The coefficient matrix is not modified by SIMEQN. Hence, changes to $\{B\}$ only allow additional solutions to be easily obtained.

CALLING SEQUENCE: SIMEQN($A(DV)$)

where the array is formatted exactly as follows:

$$N,A(1,1),A(1,2),\ldots A(N,N),B1,\ldots,BN,S1,\ldots,SN$$

CURVE FIT/TEMPERATURE DERIVATIVE

SUBROUTINE NAME: LSTSQU

PURPOSE:

This subroutine performs a least squares curve to fit to an arbitrary number of *X*, *Y* pairs to yield a polynomial equation of up to order 10. Rather than using a double precision matrix inverse, this subroutine calls on the subroutine SIMEQN to obtain a simultaneous solution.

RESTRICTIONS:

All values must be floating point numbers except *N* and *M* which must be integers. *N* is the order of the polynomial desired and is one less than the number of coefficients desired. *M* is the array length of the independent *X* or dependent *Y* values.

CALLING SEQUENCE: LSTSQU(*N, M,X(DV),Y(DV),A(DV)*)

DYNAMIC STORAGE REQUIREMENTS:

This subroutine requires 2**M* dynamic storage core locations.

SUBROUTINE NAMES: CMPXSR or CSQRI

PURPOSE:

These subroutines obtain the complex square root of a complex number or an array of complex numbers respectively. Their respective operations are:

$$A + iB = \sqrt{C + iD} \quad , \quad i = \sqrt{-1}$$

or

$$Aj + iBj = \sqrt{Cj + iDj} \quad , \quad j = 1,N$$

RESTRICTIONS:

All numbers must be floating point except *N* which must be an integer.

CALLING SEQUENCE: CMPXSR(*C,D,A,B*)

or CSQRI(*N,C(DV),D(DV),A(DV),B(DV)*)

SUBROUTINE NAMES: CMPXMP or CMPYI

PURPOSE:

These subroutines will multiply two complex numbers or the corresponding elements of arrays of complex numbers. Their respective operations are:

$$A + iB = (C + iD)*(E + iF) \quad , \quad i = \sqrt{-1}$$

$$\text{or} \quad Aj + iBj = (Cj + iDj)*(Ej + iFj) \quad , \quad j = 1,N$$

RESTRICTIONS:

All numbers must be floating point except for N which must be an integer.

CALLING SEQUENCE: CMPXMP(C,D,E,F,A,B)

or CMPYI($N,C(DV),D(DV),E(DV),F(DV),A(DV),B(DV)$)

DIVISION OPERATION

SUBROUTINE NAMES: CMPXDV or CDIVI

PURPOSE:

These subroutines will divide two complex numbers or the corresponding elements of arrays of complex numbers. Their respective operations are:

$$A + iB = (C + iD)/(E + iF) \quad , \quad j = \sqrt{-1}$$

$$\text{or} \quad Aj + iBj = (Cj + iDj)/(Ej + iFj) \quad , \quad j = 1,N$$

RESTRICTIONS:

All numbers must be floating point except for N which must be an integer.

CALLING SEQUENCE: CMPXDV(C,D,E,F,A,B)

or CDIV($N,C(DV),D(DV),E(DV),F(DV),A(DV),B(DV)$)

2.7 ARRAY OPERATIONS AND MANIPULATIONS

ADDITION OPERATION

SUBROUTINE NAMES: ADDARY or ARYADD

PURPOSE:

Subroutine ADDARY will add the corresponding elements of two specified length arrays to form a third array. Subroutine ARYADD will add a constant value to every element in an array to form a new array. Their respective operations are:

$$Ai = Bi + Ci \quad , \quad i = 1,N$$

$$\text{or} \quad Ai = Bi + C \quad , \quad i = 1,N$$

RESTRICTIONS:

All data values to be operated on must be floating point numbers. The array length N must be an integer.

CALLING SEQUENCE: ADDARY(N,$B(DV)$,$C(DV)$,$A(DV)$)

or ARYADD(N,$B(DV)$,C,$A(DV)$)

The answer array may be overlayed into one of the input array areas.

SUBROUTINE NAME: SUMARY

PURPOSE:

To sum an array of floating point values:

$$S = \Sigma\, Ai \quad , \quad i = 1,N$$

RESTRICTIONS:

The values to be summed must be floating point numbers and the array length N must be an integer.

CALLING SEQUENCE: SUMARY (N,$A(DV)$,S)

SUBTRACTION OPERATION

SUBROUTINES NAMES: SUBARY or ARYSUB

PURPOSE:

Subroutine SUBARY will subtract the corresponding elements of one array from another to form a third array. Subroutine ARYSUB will subtract a constant value from every element in an array to form a new array. Their respective operations are:

$$Ai = Bi - Ci \quad , \quad i = 1,N$$

or

$$Ai = Bi - C \quad , \quad i = 1,N$$

RESTRICTIONS:

All data values to be operated on must be floating point numbers. The array length N must be an integer.

CALLING SEQUENCE: SUBARY(N,$B(DV)$,$C(DV)$,$A(DV)$)

or ARYSUB(N,$B(DV)$,C,$A(DV)$)

The answer array may be overlayed into one of the input array areas.

MULTIPLICATION OPERATION

SUBROUTINE NAMES: MPYARY or ARYMPY

PURPOSE:

Subroutine MPYARY will multiply the corresponding elements of two arrays to form a third. Subroutine ARYMPY will multiply a constant value times each element of an array to form a new array. Their respective operations are:

$$Ai = Bi * Ci \quad , \quad i = 1,N$$

or

$$Ai = Bi * C \quad , \quad i = 1,N$$

RESTRICTIONS:

All data values to be operated on must be floating point numbers. The array length N must be an integer.

CALLING SEQUENCE: MPYARY(N,$B(DV)$,$C(DV)$,$A(DV)$)

or ARYMPY(N,$B(DV)$,C,$A(DV)$)

The answer array may be overlayed into one of the input array areas.

MULTIPLICATION OPERATION

SUBROUTINE NAMES: SCLDEP or SCLIND

PURPOSE:

These subroutines will multiply the dependent or independent variables of a doublet type interpolation array respectively. Their respective operations are:

$$Ai = X*Ai \quad , \quad i = 2,4,6,8,\ldots,n$$

or $$Ai = X*Ai \quad , \quad i = 1,3,5,7,\ldots,n-1$$

RESTRICTIONS:

All values must be floating point. The arrays must be referenced with the integer count form.

CALLING SEQUENCE: SCLDEP(*A(IC),X*)

or SCLIND(*A(IC),X*)

DIVISION OPERATION

SUBROUTINE NAMES: DIVARY or ARYDIV

PURPOSE:

Subroutine DIVARY will divide the elements of one array into the corresponding elements of another array to produce a third array. Subroutine ARYDIV will divide each element of an array by a constant value to produce a new array. Their respective operations are:

$$Ai = Bi/Ci \quad , \; i = 1,N$$

or $$Ai = Bi/C \quad , \; i = 1,N$$

RESTRICTIONS

All data values to be operated on must be floating point numbers. The array length *N* must be an integer.

CALLING SEQUENCE: DIVARY(*N,B(DV),C(DV),A(DV)*)

or ARYDIV(*N,B(DV),C,A(DV)*)

The answer array may be overlayed into one of the input array areas.

DIVISION OPERATION

SUBROUTINE NAMES: ARYINV or ARINDV

PURPOSE:

Subroutine ARYINV will invert each element of an array in its own location. Subroutine ARINDV will divide each element of an array into a constant value to form a new array. Their respective operations are:

$$Ai = 1.0/Ai \quad , \quad i = 1,N$$

or

$$Ai = B/Ci \quad , \quad i = 1,N$$

RESTRICTIONS:

All data values must be floating point numbers. The array length N must be an integer.

CALLING SEQUENCE: ARYINV(N,$A(DV)$)

or ARINDV(N,$C(DV)$,B,$A(DV)$)

(The ARINDV answer array may be overlayed into the input array area.)

SUBROUTINE NAME: ADARIN

PURPOSE:

Subroutine ADARIN will calculate one over the sum of inverses of an array of values. This subroutine is useful for calculating the effective conductance of series conductors. The operations are:

$$Y = 1.0/\Sigma(1./Xi) \quad , \quad i = 1,2,\ldots, N$$

RESTRICTIONS:

All data values must be floating point numbers. The array length N must be an integer.

CALLING SEQUENCE:

ADARIN (N, $X(DV)$, Y)

DISTRIBUTION OF ARRAY DATA

SUBROUTINE NAMES: SHFTV or SHFTVR or FLIP

PURPOSE:

Subroutine SHFTV will shift a sequence of data from one array to another. Subroutine SHFTVR will shift a sequence of data from one array and place it in another array in reverse order. Subroutine FLIP will reverse an array in its own array location. Their respective operations are:

$$A(i) = B(i) \quad , \quad i = 1,N$$

or $$A(N-i+1) = B(i) \quad , \quad i = 1,N$$

or $$A(i)new = A(n-i+2)old \quad , \quad i = 2,n+1$$

RESTRICTIONS:

The data values to be shifted or reversed in order may be anything. The N must be an integer.

CALLING SEQUENCE: SHFTV(N,$B(DV)$,$A(DV)$)

or SHFTVR(N,$B(DV)$,$A(DV)$)

or FLIP($A(IC)$)

The answer array may not be overlayed into the input array.

SUBROUTINE NAME: GENARY

PURPOSE:

This subroutine will generate an array of equally incremented ascending values. The user must supply the minimum value, maximum value, number of values in the array to be generated and the space for the generated array.

RESTRICTIONS:

All numbers must be floating point.

CALLING SEQUENCE: GENARY($B(DV)$,$A(DV)$)

where $B(1)$ = minimum value

$B(2)$ = maximum value

$B(3)$ = length of array to be generated (floating point)

DISTRIBUTION OF ARRAY DATA

SUBROUTINE NAME: BLDARY

PURPOSE:

This subroutine will build an array from a variable number of arguments in the order listed. The operation performed is:

$$Ai = Xi \quad , \quad i = 1,n$$

RESTRICTIONS:

Data may be of any form. The subroutine obtains the integer array length n by counting the arguments.

CALLING SEQUENCE: BLDARY(*A(DV),X1,X2,X3,...,Xn*)

SUBROUTINE NAME: BRKARY or BKARAD

PURPOSE:

These subroutines will distribute values from within an array to a variable number of arguments in the order listed. The first places the value into the location while the second adds it to what is in the location. Respective operations are:

$$Xi = Ai \quad , \quad i = 1,n$$

or $$Xi = Xi + Ai \quad , \quad i = 1,n$$

RESTRICTIONS:

Floating point numbers must be used for BKARAD. The integer array length n is obtained by the routines by counting the number of arguments.

CALLING SEQUENCE: BRKARY(*A(DV),X1,X2,X3,...,Xn*)

or BKARAD(*A(DV),X1,X2,X3,...,Xn*)

SUBROUTINE NAMES: STØARY or ARYSTØ

PURPOSE:

These subroutines will place a value into or take a value out of a specific array location respectively. Their respective operations are:

$$Ai = X \quad , \quad i = N \quad , N > 0$$

or $$X = Ai \quad , \quad i = N \quad , N > 0$$

RESTRICTIONS:

The value may be anything but N must be an integer.

CALLING SEQUENCE: STØARY(*N,X,A(DV)*)

or ARYSTØ(*N,X,A(DV)*)

SUBROUTINE NAMES: STFSEQ or STFSQS

PURPOSE:

Both subroutines will stuff a constant data value into a specified length array or group of sequential locations. STFSEQ expects the constant data value to be in the first array location while STFSQS requires it to be supplied as an additional argument. The respective operations performed are:

$$Ai = A1 \quad , \quad i = 2,N$$

or

$$Ai = B \quad , \quad i = 1,N$$

RESTRICTIONS:

N must be an integer but the constant data value may be integer, floating point or alpha-numeric.

CALLING SEQUENCE: STFSEQ(*A(DV)*,*N*)

or STFSQS(*B*,*N*,*A(DV)*)

DISTRIBUTION OF ARRAY DATA

SUBROUTINE NAMES: SLDARY or SLDARD

PURPOSE:

These subroutines are useful for updating fixed length interpolation arrays during a transient analysis. The array data values are moved back one or two positions, the first one or two values discarded and the last one or two values updated respectively. The "sliding array" thus maintained can then be used with standard interpolation subroutines to simulate transport delay phenomena. Their respective operations are:

$$Ai = Ai+1 \quad , \quad i = 2,N$$

and $Ai = X \quad , \quad i = N + 1$

or $Ai = Ai+2 \quad , \quad i = 2,N-1$

and $Ai = X$ and $Ai+1 = Y \quad , \quad i = N$

RESTRICTIONS:

The addressed arrays must have the array integer count N as the first value. For SLDARD, N must be even.

CALLING SEQUENCE: SLDARY(*X,A(IC)*)

SLDARD(*X,Y,A(IC)*)

SUBROUTINE NAME: STØRMA

PURPOSE:

This subroutine is useful for constructing historical data arrays during a transient analysis. It can take the place of several STØARY calls. The operations are as follows:

$$\begin{aligned} A1(N) &= X1 \\ A2(N) &= X2 \\ A3(N) &= X3 \\ &\vdots \end{aligned}$$

RESTRICTIONS:

N must be or reference an integer, the *X*'s may be any value.

CALLING SEQUENCE: STØRMA(*N,X1,A1(DV),X2,A2(DV),X3,A3(DV),...*)

SINGLET/DOUBLET ARRAY GENERATION

SUBROUTINE NAMES: SPLIT or JØIN

PURPOSE:

These subroutines separate a doublet array into two singlet arrays or combine two singlet arrays into a doublet array respectively. Their respective operations are:

$$\begin{aligned} & Bi = A2i\text{-}1 && , && i = 1,N \\ & Ci = A2i && , && i = 1,N \\ \text{or} \quad & A2i\text{-}1 = Bi && , && i = 1,N \\ & A2i = Ci && , && i = 1,N \end{aligned}$$

RESTRICTIONS:

The arrays may contain any values but N must be an integer. N is the length of the B and C arrays and the A array must be the length of $2N$.

CALLING SEQUENCE: SPLIT(N,$A(DV)$,$B(DV)$,$C(DV)$)

or JØIN(N,$B(DV)$,$C(DV)$,$A(DV)$)

SUBROUTINE NAME: SPREAD

PURPOSE:

This subroutine applies interpolation subroutine D1D1DA to singlet arrays to obtain an array of dependent variables versus an array of independent variables. It is extremely useful for obtaining singlet arrays of various dependent variables with a corresponding relationship to one singlet independent variable array. The dependent variable arrays thus constructed can then be operated on by array manipulation subroutines in order to form composite or complex functions. Doublet arrays can first be separated with subroutine SPLIT and later reformed with subroutine JØIN.

RESTRICTIONS:

All data values must be floating point except N which must be the integer length of the array to be constructed. The arrays fed into D1D1DA for interpolation must start with the integer count. X is for independent and Y is for dependent. I is for input and $\emptyset$ is for output.

CALLING SEQUENCE: SPREAD(N,$X(IC)$,$Y(IC)$,$XI(DV)$,$Y\emptyset(DV)$)

COMPARISON OPERATION

SUBROUTINE NAMES: MAXDAR or MXDRAL

PURPOSE:

These subroutines will obtain the absolute maximum difference between corresponding elements of two arrays of equal length N. The array values must be floating point numbers. The operation performed is

$$D = \left| Ai - Bi \right|_{max} , \quad i = 1,N$$

Subroutine MXDRAL also locates the position P between 1 and N where the maximum occurs.

RESTRICTIONS:

The N argument must be an integer. The D and P arguments are returned as floating point numbers.

CALLING SEQUENCE: MAXDAR(N,$A(DV)$,$B(DV)$,D)

or MXDRAL(N,$A(DV)$,$B(DV)$,D,P)